ОПЕРАЦИОННАЯ СИСТЕМА

ОСРВ СМ ЭВМ

ОПЕРАЦИОННАЯ СИСТЕМА ОСРВ СМ ЭВМ

СПРАВОЧНОЕ ИЗДАНИЕ

МОСКВА
"ФИНАНСЫ И СТАТИСТИКА"
1987

ББК 32.973
О-60
УДК 681.3.06

Рецензент **Ю. П. Шакула**
Авторы:
Г. А. Егоров, В. Л. Кароль, И. С. Мостов, М. Б. Немировский, А. П. Пиядин, Л. Н. Столяр, В. А. Шапошников, Л. Ф. Штильман

О-60 **Операционная** система ОСРВ СМ ЭВМ: Справочное издание/Г. А. Егоров, В. Л. Кароль, И. С. Мостов и др.— М.: Финансы и статистика, 1987.—271 с.: ил.

Операционная система реального времени (ОСРВ) СМ ЭВМ — наиболее распространенная система для ЭВМ этого класса. В книге излагаются структура, состав и функции ОСРВ Основное внимание уделяется управляющим программам, организации файловой системы и системы ввода-вывода, а также системным обслуживающим задачам.

Для системных и проблемных программистов — пользователей СМ ЭВМ, а также для инженеров и научных работников, занимающихся разработкой различных систем управления на базе СМ ЭВМ. Может быть полезна аспирантам и студентам вузов.

$О\ \frac{2405000000-051}{010(01)-87}\ 108-87$ ББК 32.973

© Издательство «Финансы и статистика», 1987

ВВЕДЕНИЕ

Операционная система реального времени (ОСРВ) – наиболее распространенная из операционных систем СМ ЭВМ [1]. Она предназначена для использования в качестве базовой системы в различных применениях реального времени. Развитые средства организации многопользовательского режима работы, управления файлами на различных магнитных накопителях, наличие широкого набора обслуживающих программ позволяют использовать ОСРВ в качестве универсальной инструментальной системы для разработки и отладки программ. Этому способствует широкий набор систем программирования, входящих в состав ОСРВ: МАКРО, Фортран, Кобол, Паскаль, Бейсик.

Система ОСРВ характеризуется эффективным использованием вычислительных ресурсов, имеет широкий набор управляющих функций и большой выбор возможностей работы с внешними устройствами. В зависимости от требований к составу технических средств целевого комплекса и выбираемых возможностей могут генерироваться различные варианты операционной системы.

Разработан ряд программных средств телеобработки, функционирующих под управлением ОСРВ: программное обеспечение для организации однородных сетей СМ ЭВМ [2] и пакеты программ для построения неоднородных многомашинных комплексов на базе СМ ЭВМ и ЕС ЭВМ [3]. Для работы с базами данных под управлением ОСРВ могут использоваться пакеты системных прикладных программ ФОБРИН, МИРИС [4].

ОСРВ включает развитые программные средства, обеспечивающие проверку работоспособности вычислительного комплекса (ВК) и повышение надежности его функцио-

нирования. Это обеспечивается за счет реализации управления динамической памятью, резервирования дисков, организации виртуального доступа из одного ВК к вычислительным ресурсам другого и т. п.

Все описанные функции наряду с широким набором системных средств, предоставляемых пользователям для реализации проблемных применений, обеспечивают использование ОСРВ в различных областях.

Предлагаемая книга представляет собой справочное издание и предназначается для системных и проблемных программистов-пользователей СМ ЭВМ, а также для инженеров и научных работников, занимающихся разработкой различных систем управления на базе СМ ЭВМ. Она может быть полезна также аспирантам и студентам вузов.

1

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ОСРВ

ОСРВ — одна из самых распространенных операционных систем реального времени для СМ ЭВМ с интерфейсом «Общая шина». С 1978 г. выпущено несколько версий ОСРВ. В справочнике описываются возможности основных компонентов третьей версии ОСРВ, выпущенной в 1984 г.

1.1. ОБЛАСТЬ ПРИМЕНЕНИЯ ОСРВ

Операционная система реального времени ОСРВ может использоваться на различных моделях семейства СМ ЭВМ с магистральной архитектурой, т. е. имеющих интерфейс «Общая шина». В класс ЭВМ, поддерживаемых ОСРВ, входят СМ-3, СМ-4, СМ1420, СМ1600, СМ1300 и другие программно совместимые с ними мини-ЭВМ, выпускаемые в нашей стране и за рубежом.

Основным назначением ОСРВ является организация вычислительного процесса в режиме реального времени. ОСРВ обеспечивает быструю реакцию на события и ввод данных, включает развитые средства мультипрограммирования, широкие возможности для многотерминальной работы, поддержку разнообразного оборудования, входящего в номенклатуру СМ ЭВМ. Сочетание этих свойств, гибкость компоновки комплексов и генерации системы обеспечивают обширную область проблемных применений управляющих вычислительных комплексов СМ ЭВМ, функционирующих под управлением ОСРВ.

Типичные случаи применения ОСРВ:

сбор и переработка информации в реальном масштабе времени;

управление технологическими объектами;

информационно-справочные системы;

многотерминальная работа в режиме разделения времени;

распределенная телеобработка.

Развитые системные средства управления мультизадачной средой позволяют в одной ЭВМ сочетать различные сферы применения с режимом диалоговой разработки и отладки программ.

Широкому применению комплексов СМ ЭВМ, работающих под управлением ОСРВ, способствует большой выбор пакетов прикладных программ (ППП), предназначенных для организации баз данных и распределенной обработки данных.

Для организации баз данных имеются следующие пакеты:

система управления данными СУД2,

диалоговая система ФОБРИН2,

база данных сетевой структуры СЕТОР,

база данных реляционного типа МИРИС,

набор подпрограмм на языке Бейсик-П2 РИС для организации реляционных баз данных.

В ОСРВ система СУД2 входит в качестве стандартного компонента подсистемы логического ввода-вывода. Остальные перечисленные ППП непосредственно в состав системы ОСРВ не входят и поставляются отдельно.

Для организации многомашинных комплексов разработан ряд пакетов, функционирующих под управлением ОСРВ. Для организации локальных однородных многомашинных комплексов СМ ЭВМ может применяться система виртуального доступа (СВД), которая организует «прозрачный» доступ к оборудованию локально связанных (через контроллер К5СМ — дуплексный регистр) установок.

Для создания однородных распределенных многомашинных комплексов СМ ЭВМ может использоваться пакет программ сетевой телеобработки ПП СТО/РВ и его развитие пакет «СЕТЬ СМ». Эти пакеты обеспечивают взаимодействие удаленных ЭВМ и передачу данных через аппаратуру синхронной и асинхронной связи (БС АДС, АДС-С, МПД-ПСА и др.).

Для организации неоднородных многомашинных комплексов на базе СМ ЭВМ и ЕС ЭВМ могут использоваться

пакеты ПП ММК/Л и ПП ММК/Р, обеспечивающие локальное и удаленное взаимодействие комплексов через УСВМ и АДС-С.

ОСРВ является основной частью базового программного обеспечения (БПО) ряда проблемно ориентированных комплексов (ПОК), построенных на основе типовых комплексов СМ ЭВМ и специализированного внешнего оборудования. Это измерительные вычислительные комплексы (ИВК) для автоматизации научных исследований с аппаратурой КАМАК и АСЭТ, обеспечивающей подключение к приборам. На основе ОСРВ построено БПО автоматизированных рабочих мест (АРМ) — комплексов для автоматизации проектно-конструкторских работ. АРМы построены на основе моделей СМ-3 и СМ-4, оснащенных устройствами для ввода и вывода графической информации.

Для комплексов повышенной производительности ОСРВ обеспечивает двухпроцессорную работу с общей двухвходовой памятью.

Для двухмашинных функционально распределенных комплексов СМ1420, объединенных через переключатели шины и дуплексные регистры, ОСРВ обеспечивает средства резервирования дисков, возможность реконфигурации комплекса и совместное использование оборудования через систему виртуального доступа (СВД).

Многообразие типовых проблемно-ориентированных комплексов, работающих под управлением ОСРВ, постоянно растет, что обеспечивает новые области применения.

1.2. СТРУКТУРА, СОСТАВ И ФУНКЦИИ ОСРВ

Операционная система ОСРВ является комплексом программ, обеспечивающим организацию вычислительного процесса. Она состоит из следующих компонентов

ядра операционной системы;

набора системных управляющих программ;

набора системных обслуживающих программ;

системы программирования, средства создания и отладки задач;

средств генерации системы;

средств повышения надежности функционирования.

Некоторые из перечисленных компонентов ОСРВ описаны в справочнике подробно, другие ввиду ограниченности объема справочника только упомянуты.

1.2.1. ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ

Компоненты ядра ОСРВ являются резидентными в оперативной памяти. Они обеспечивают управление оперативной памятью, распределение ресурсов центрального процессора (диспетчеризация задач и обслуживание запросов к управляющей программе) обработку прерываний и другие функции.

Функции, реализуемые управляющей программой, описаны в п. 1.3, запросы задач (директивы) к управляющей программе — в гл. 2.

1.2.2. СИСТЕМНЫЕ УПРАВЛЯЮЩИЕ ПРОГРАММЫ

Набор системных управляющих программ обеспечивает средства управления данными, взаимодействие с оператором, управление системой.

Средства управления данными ОСРВ подразделяются на систему управления файлами FCS и систему управления данными СУД. FCS обеспечивает последовательный и прямой методы доступа. Глава 3 содержит подробное описание системы управления файлами ОСРВ. СУД обеспечивает три способа организации файлов: последовательный, относительный и индексный. Средства СУД позволяют программам совместно обращаться к записям файлов и обеспечивают возможность обработки нескольких запросов к одному файлу, синхронную и асинхронную обработку записей, обработку записей в режиме пересылки или указания места. Средства СУД в справочнике не описываются.

Взаимодействие пользователей с системой реализуется программой связи с оператором MCR, которая обеспечивает интерактивное взаимодействие пользователей с системой. Программы связи с оператором описаны в гл. 4.

Набор задач управления системой обеспечивает средства регистрации пользователей в системе (задача ACNT), протоколирования работы системы (задача COLOG), процедуры завершения работы системы (задача SHUTUP), средства управления системным выводом (задачи QMG, PRT).

1.2.3. СИСТЕМНЫЕ ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ

Системные обслуживающие программы можно разбить на группы в соответствии с их функциональным назначением: программы работы с файлами и томами; программы редактирования и форматирования текстов; программа работы с библиотеками и программы корректировки объектных модулей и образов задач.

Подробное описание работы с системными обслуживающими программами приведено в гл. 6. Группа программ работы с файлами и томами включает:

программу работы с файлами PIP, обеспечивающую работу с каталогами, файлами (перепись, слияние, переименование, установку защиты файлов, удаление, разблокирование, распечатку и др.);

программу распечатки файлов DMP, которая позволяет вывести содержимое файла или указанных блоков диска на выводное устройство в различных форматах (восьмеричном, символьном, RAD-50, шестнадцатеричном и т. д.);

программу преобразования файлов FLX, которая производит обслуживание томов и передачу файлов между томами с файловой структурой ОСРВ, ДОС СМ, ФОБОС и РАФОС;

программу сравнения файлов CMP, которая производит сравнение символьных файлов;

программу копирования и уплотнения тома DSC, позволяющую сохранить немонтированный дисковый том на магнитной ленте или другом диске. При операции копирования дискового тома производится уплотнение файлов с целью устранения фрагментации свободного пространства;

программу копирования и восстановления томов BRU, которая позволяет сохранить и восстановить тома или отдельные файлы, а также проверить их идентичность;

программу форматирования дисков FMT, позволяющую под управлением системы произвести форматирование дискового тома и проверить корректность заголовков секторов;

программу проверки томов BAD, которая позволяет обнаружить дефектные блоки на дисковом томе и исключить их из дальнейшей работы;

программу проверки файловой структуры VFY, позволяющую проверить корректность файловой структуры и данных на диске, обнаружить и идентифицировать некорректные файлы;

программу сохранения томов PRESRV, обеспечивающую копирование магнитных лент и дисков и восстановление томов с файловой структурой ОСРВ или с другой файловой структурой;

программу управления магнитной лентой MAG, которая позволяет задавать атрибуты файла на магнитной ленте без меток, обеспечивает позиционирование лент без меток и лент со стандартными метками. При монтировании магнитной ленты можно указать потребность в перекодировке, например из кода ДКОИ во внутренний код СМ ЭВМ для лент, подготовленных на ЕС ЭВМ.

Группа программ редактирования и форматирования текстов включает:

диалоговый редактор EDI, позволяющий осуществить редактирование строк символьного файла (поиск, замену, удаление и другие операции) при работе за терминалом;

пакетный редактор SLP, обеспечивающий редактирование строк символьных файлов (удаление строк, вставку, индикацию изменений и др.);

диалоговый экранный редактор текстов RETEX, позволяющий производить при работе за терминалом ввод, экранное редактирование, форматирование документов для вывода в одну или две колонки, нумерацию страниц и постраничную распечатку. RETEX генерируется в зависимости от типа терминала и параметров форматирования текстов;

универсальный редактор EDT, обеспечивающий работу в строковом и экранном режимах с возможностью использования функциональной клавиатуры;

программу форматирования текстов DOC, позволяющую выполнить форматирование предварительно подготовленного описания документа, осуществить разбиение на страницы, сбор оглавления, распечатку в одну или две колонки;

программируемый редактор текста TECO, позволяющий производить операции диалогового редактирования произвольных текстов с использованием команд поиска, выделения, разделения, слияния, сохранения и т. п., а также с возможностью написания процедур редактирования с учетом специфики текста.

Программа работы с библиотеками LBR позволяет формировать и модифицировать следующие библиотеки: библиотеку макроопределений, библиотеку объектных модулей и универсальную библиотеку.

В отличие от библиотеки макроопределений и объектных модулей универсальная библиотека может содержать однотипные модули произвольного вида. LBR обеспечивает включение модулей в библиотеку, удаление модулей и точек входа, уплотнение библиотеки, выделение модуля из библиотеки, расширение пространства библиотеки с изменением области оглавления, распечатку содержимого библиотеки.

В группу программ корректировки включены:

программа корректировки объектных модулей PAT, которая позволяет корректировать объектный модуль на основе заранее подготовленных и оттранслированных изменений к нему;

программа корректировки образов задач ZAP, которая позволяет внести корректировки в файл образа задачи на диске.

1.2.4. СРЕДСТВА СОЗДАНИЯ И ОТЛАДКИ ЗАДАЧ

ОСРВ имеет набор трансляторов со следующих языков программирования: макроассемблер, Фортран IV, Кобол, Паскаль, Бейсик и Бейсик-П2. Эти системы программирования включают трансляторы и соответствующие им объектные библиотеки исполняющих систем. Исключение составляет интерпретатор с языка Бейсик. Описание систем программирования не включено в справочник.

Для создания выполняемой задачи ее текст на исходном языке следует обработать соответствующим транслятором и затем Построителем задач ТКВ, который описан в гл. 5. При построении задачи в нее могут быть включены средства отладки. ОСРВ обеспечивает диалоговый отладчик ODT и трассирующий отладчик TRACE.

1.2.5. СРЕДСТВА ГЕНЕРАЦИИ СИСТЕМЫ

Настройка операционной системы ОСРВ в соответствии с требованиями конкретного проблемного применения производится в результате выполнения действий, называемых *генерацией системы*. Генерация требуемого варианта ОСРВ производится под управлением рабочей системы путем выполнения командных файлов генерации, находящихся на дистрибутивных носителях системы.

В процессе выполнения командных файлов генерации создаются файлы определения переменных условной

трансляции модулей управляющей программы; таблицы, отражающие конфигурацию оборудования; командные файлы для трансляции и компоновки модулей управляющей программы, для компоновки привилегированных и непривилегированных системных задач.

Полная генерация системы выполняется в три фазы. В первой фазе определяется конфигурация системы и транслируются модули управляющей программы. Во второй фазе строится образ управляющей программы и привилегированные задачи. В третьей фазе строятся непривилегированные задачи. Для получения работоспособной целевой системы образ управляющей программы должен быть настроен. Виртуальная программа связи с оператором VMR осуществляет распределение памяти для разделов в образе системы, установку необходимых задач, загрузку драйверов и другие необходимые действия, после которых целевая система может быть загружена и инициирована. Загруженная целевая система может быть модифицирована в процессе функционирования командами программы связи с оператором MCR.

1.2.6. СРЕДСТВА ПОВЫШЕНИЯ НАДЕЖНОСТИ ФУНКЦИОНИРОВАНИЯ

ОСРВ включает набор программных средств, обеспечивающих повышение надежности и устойчивости функционирования комплекса. Средства повышения надежности обеспечивают следующие функции.

Устойчивость к ошибкам оборудования. В случае возникновения ошибки при передаче данных драйвер (программа управления устройством) пытается ее скорректировать. При обнаружении ошибки значения регистров устройства передаются подсистеме регистрации ошибок оборудования ERRLOG, которая накапливает статистику программно корректируемых и некорректируемых ошибок. Анализ статистики ошибок позволяет вóвремя выполнять профилактический ремонт и тем самым избежать разрушения и потери данных. Средства регистрации и анализа ошибок оборудования описаны в гл. 8.

Проверка работоспособности оборудования. Работоспособность оборудования может быть проверена под управлением ОСРВ средствами пакетов ЭКСПЕРТ и ПРОРАБ (гл. 8).

Пакет ЭКСПЕРТ содержит задачи-тесты для каждого типа проверяемого устройства. При проверке накопителей

на магнитных лентах и дисках происходит разрушение записанной на них информации. Пакет ПРОРАБ позволяет проверить работоспособность накопителя без разрушения информации, записанной на томе.

Резервирование дисков. ОСРВ включает средства программного резервирования дисковых томов. Работа с резервированными дисками для прикладных задач аналогична работе с обычным диском. На уровне генерации системы группа однотипных дисковых устройств может быть объявлена резервной. Запись данных осуществляется на все диски группы, а чтение — с одного из дисков. При чтении выбирается диск, с которого информация считывается без ошибок. Устройство, на котором обнаружена ошибка, автоматически выводится из резервирования. Информация на диске автоматически восстанавливается через некоторое время после повторного включения диска в группу резервных.

Обеспечение устойчивой работы системы. Управляющая программа ОСРВ обеспечивает автоматическое слежение за состоянием системной динамической области памяти (пула).

При перегрузке системы задача управления пулом РМТ ограничивает активность задач с приоритетом ниже заданного, запрещает регистрацию новых пользователей, информирует оператора о перегрузке системы.

Консольное протоколирование. ОСРВ имеет средства протоколирования сообщений, выводимых на псевдоустройство СО:. Это позволяет регистрировать последовательность событий, происходящих в системе. Когда процесс консольного протоколирования активен, вывод на СО: передается задаче СОТ для пересылки сообщения в файл протокола или на терминал. СОТ добавляет к сообщению метку, содержащую текущее время, что позволяет выделить сообщения, выведенные за определенный интервал времени.

Анализ аварийного состояния. При аварии системы содержимое оперативной памяти может быть сохранено на диске или магнитной ленте программой сохранения аварийного состояния.

Программа анализа аварийного состояния CDA позволяет выдать в удобном формате сохраненное состояние памяти системы и содержимое системных структур данных. Полученные данные являются основой для анализа причин сбоев и отказов системы. Программа описана в гл. 8.

Авторизация доступа к системе. Доступ к многопользовательской системе разрешается после успешной реги-

страции. В процессе регистрации запрашиваются имя, пароль и права доступа, установленные администратором системы с помощью задачи ACNT, которая создает и ведет файл учетной информации с записями о каждом пользователе. Этот файл содержит:

код идентификации пользователя UIC, определяющий привилегии доступа;
системное устройство по умолчанию;
имя и пароль;
имя интерпретатора командных строк;
дату и время последней регистрации;
счетчик регистрации;
состояние терминала (подчиненность).

1.3. ВОЗМОЖНОСТИ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

Управляющая программа ОСРВ реализует следующие основные функции: управление оперативной памятью; управление задачами; обслуживание запросов задач к управляющей программе и обслуживание прерываний.

Единицей выполняемой работы в системе является *задача.* Атрибуты задачи хранятся в заголовке задачи и блоке управления задачи TCB, форматы которых приведены в приложении. Атрибуты задачи задаются построителем задач (гл. 5) и могут изменяться при установке задачи (гл. 4). Они определяют характеристики задачи: привилегированность, подчиненность, выгружаемость и др. Подчиненная задача получает привилегии доступа к файлам от запустившей ее задачи.

Задачи могут быть *привилегированными* и *непривилегированными.* Привилегированной задаче доступна область памяти управляющей программы и внешняя страница памяти, где размещаются регистры внешних устройств.

1.3.1. УПРАВЛЕНИЕ ОПЕРАТИВНОЙ ПАМЯТЬЮ

В оперативной памяти располагаются: управляющая программа (ядро операционной системы), задачи пользователей и различные структуры данных, описывающие состояние системы. Оперативная память системы делится на разделы.

Раздел представляет собой непрерывную область оперативной памяти и имеет следующие характеристики: имя, базовый адрес, длину и тип.

Имя раздела включает до шести буквенно-цифровых символов (в латинском алфавите). *Базовый адрес* определяет положение начала раздела в физической памяти (в блоках по 64 байта). Возможные *типы* разделов зависят от модели процессора. Архитектура младших моделей СМ ЭВМ допускает адресацию 32К 16-разрядных слов (СМ-3, СМ 1300). В старших моделях СМ ЭВМ (СМ-4, СМ 1420, СМ 1600) задача выполняется в области адресов от 0 до 32 Кслов, которые называются *виртуальными.* Устройство диспетчер памяти позволяет отобразить виртуальные адреса задач в адреса физической памяти, размер которой зависит от модели ЭВМ (128 Кслов для СМ-4 и 1920 Кслов для СМ 1420).

ОСРВ может быть сгенерирована в двух вариантах — для ЭВМ с диспетчером памяти (ДП) или без ДП. В системе с ДП задача может выполняться в любом из разделов достаточной длины. В системе без ДП задача привязывается к физическому адресу памяти и должна выполняться именно в том разделе, для которого она создана.

В ОСРВ используются разделы следующих типов: системно управляемые; управляемые пользователем; общие и разделы регистров устройств.

Системно управляемые разделы допустимы только в системе с ДП. Память такого раздела динамически распределяется системой для размещения в нем наибольшего числа задач. Аппаратура диспетчера памяти обеспечивает размещение задач независимо от физических адресов раздела, а также защиту памяти задач и управляющей программы. Система может перемещать задачи внутри системно управляемого раздела с целью устранения фрагментации памяти и загрузки ожидающих задач.

Раздел, управляемый пользователем, может быть создан в системе с ДП и системе без ДП. В разделе, управляемом пользователем, в каждый момент времени может находиться только одна задача. Раздел, управляемый пользователем, может быть разбит на несколько (до 7) непересекающихся подразделов. Несколько задач могут находиться одновременно в разных подразделах либо одна большая задача может занимать раздел целиком.

Диспетчер памяти позволяет отобразить виртуальное пространство задачи (32 Кслов) на несколько (до 8)

участков физической памяти (по 4 Кслова). Используя эту возможность, ОСРВ позволяет организовать *общие разделы* COMMON. Задача может отображать часть своего виртуального пространства в общий раздел. Отображая часть памяти двух или более задач в общий раздел, можно организовать их взаимодействие. Другая возможность — разместить в общем разделе набор подпрограмм, общих для нескольких задач (разделяемую библиотеку подпрограмм) и тем самым сэкономить оперативную память. Общий раздел может быть объявлен доступным только для чтения или для чтения и записи. Отображение части пространства задачи в общий раздел можно произвести статически — в момент создания задачи или динамически — по запросу к системе при выполнении задачи.

Раздел регистров устройств DEV предназначен для адресации внешней страницы памяти, содержащей регистры внешних устройств.

Логическое и виртуальное адресные пространства задачи. В системе без ДП виртуальное и физическое адресные пространства совпадают. В системе с ДП в каждый момент времени отображается до 32 Кслов виртуальной памяти в физическую. Эта отображаемая область виртуальной памяти называется *виртуальным адресным пространством* (ВАП). ДП выполняет отображение с помощью восьми внутренних регистров, каждый из которых позволяет отобразить от 32 слов до 4 Кслов виртуальной памяти в физическую. Непрерывное ВАП может отображаться в максимум восемь областей физической памяти (не обязательно смежных). Каждая область отображается с помощью соответствующего регистра диспетчера памяти. Адрес отдельной области и ее длина должны быть кратны 32-м словам. Рис. 1 иллюстрирует пример отображения ВАП в физическое адресное пространство (ФАП). ОСРВ имеет средства управления отображением ВАП в ФАП. Система не позволяет отображать виртуальные адреса задач в произвольную область ФАП. Память, в которую допустимо отобразить ВАП задачи, называется *логическим адресным пространством задачи.* Логическое адресное пространство задачи включает один или несколько районов, каждый из которых располагается в одном из разделов (рис. 2, с. 18).

В ОСРВ различают три типа районов:

район задачи (область памяти, в которой находятся заголовок, стек и корневая часть задачи);

статический общий район (включает память раздела типа COMMON или DEV);

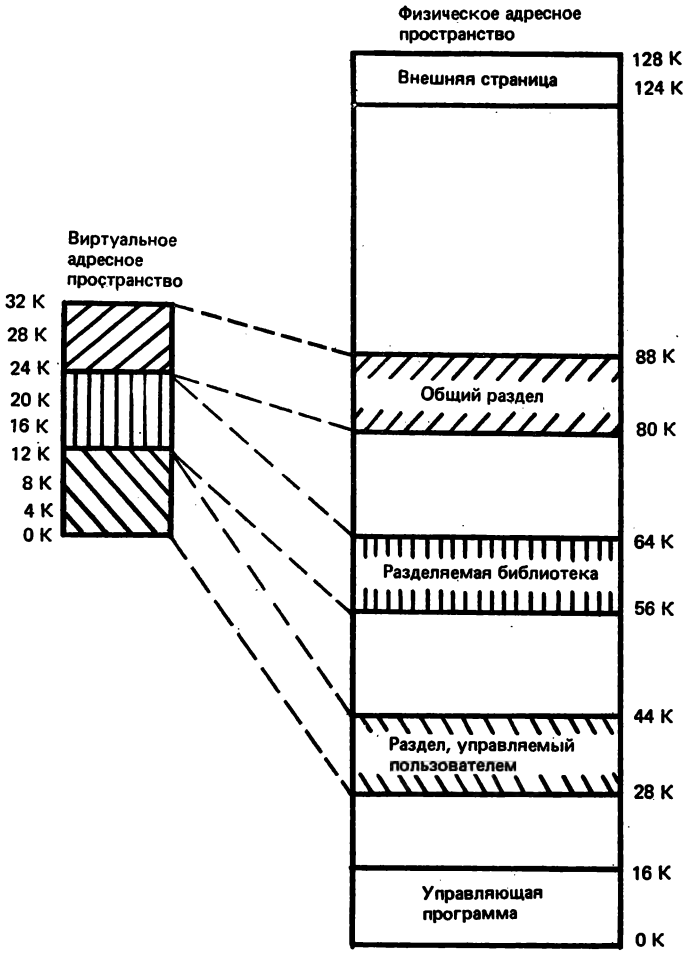


Рис. 1. Отображение виртуального адресного пространства

динамический район (создается динамически в системно-управляемом разделе посредством директивы CRRG⊙).

Задача может присоединиться к существующему району в момент запуска или по запросу к управляющей программе. Динамически созданный район может уничтожаться по запросу задачи или автоматически после завершения всех присоединяющихся к нему задач.

Для отображения виртуального пространства задачи в логическое пространство используются *виртуальные адресные окна.* Каждое адресное окно отображает непре-

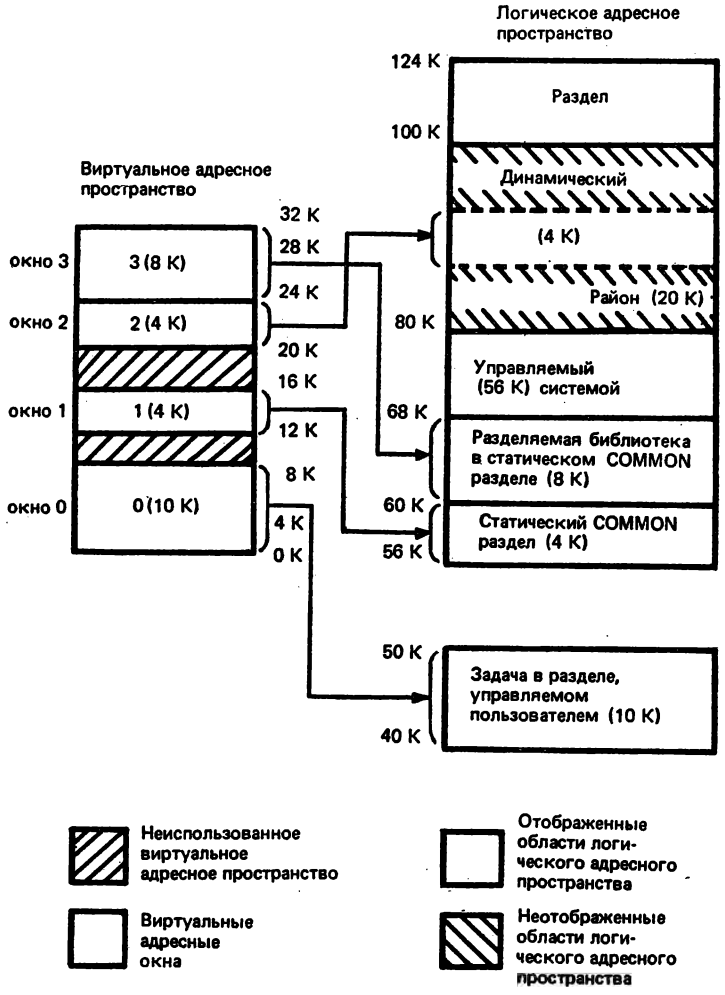


Рис. 2. Отображение виртуальных окон в районы

рывный участок виртуальных адресов, начиная с адреса, кратного 4 Кословам и длиной, кратной 32-м словам. Максимальная длина — 32 Кслова.

При запуске задачи адресные окна описывают отображение, заданное построителем задач (адресное окно всегда описывает район задачи). Если задача не использует разделяемых библиотек или общих областей, адресное окно задачи является единственным. Число возможных

адресных окон (максимум 7) задается при создании задачи. Каждое адресное окно, за исключением нулевого, может переотображаться посредством соответствующих директив.

Необходимо отметить отсутствие прямого соответствия между номером адресного окна и его базовым виртуальным адресом и то, что отображение через окно может разрушать отображение других адресных окон.

Работа с резидентными в памяти перекрытиями, описанная в гл. 5, построена на использовании директив отображения виртуальных адресных окон.

1.3.2. УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ЗАДАЧ

В каждый момент времени в системе может выполняться только одна задача. Мультипрограммирование становится возможным потому, что выполнение задачи связано не только с использованием процессора. Задача реального времени, вызывая какое-то действие и ожидая его завершения, может не занимать процессор на время ожидания. Поэтому, если хотя бы одна задача ждет завершения какого-то события (например, завершения операции ввода-вывода), то управляющая программа разрешает другой задаче использовать процессор. В ОСРВ управляющая программа координирует выполнение всех задач, находящихся в памяти, и обеспечивает эффективное использование ресурсов системы.

Для выполнения в среде ОСРВ задача должна быть установлена. *Установка задачи* — это процедура, делающая задачу известной управляющей программе. При установке задачи система записывает ряд ее параметров (имя, размер задачи, адрес на диске, с которого начинается образ задачи, и имя раздела, в котором выполняется задача) в блок управления задачей ТСВ. Блок управления задачей включается в каталог установленных задач.

Установленная задача не загружается в раздел и не конкурирует с другими задачами за системные ресурсы. Управляющая программа считает ее бездействующей до тех пор, пока не будет выдан запрос, требующий активизации установленной задачи.

Управляющая программа распознает два состояния задачи: состояние бездействия и состояние активности. *Бездействующая задача* — это задача, установленная в системе (имеется запись в STD), но не имеющая запроса на выполнение. *Активная задача* — это задача, установ-

ленная в системе и имеющая запрос на выполнение. Задача остается активной до ее завершения. После завершения задача перейдет в состояние бездействия.

В ОСРВ различают три состояния активной задачи: готова к выполнению, блокирована и остановлена. *Готовая к выполнению задача* конкурирует с другими задачами за процессор на основе приоритетов. Задача с высшим приоритетом получает право использовать процессор и становится текущей. *Задача блокирована,* если она не способна конкурировать за время процессора по причине синхронизации или занятости необходимых ей ресурсов. Приоритет блокированной задачи остается неизменным. *Остановленная задача,* как и блокированная, не конкурирует за процессор, но в отличие от блокированной остановленная задача имеет нулевой приоритет и может быть выгружена любой задачей.

Различие между бездействующими и активными задачами является важным в системах реального времени. Задача в состоянии бездействия занимает меньше памяти, чем в состоянии активности, но когда она потребуется для обслуживания событий реального времени, управляющая программа может быстро и эффективно ввести ее в активную конкуренцию за системные ресурсы.

Запись в STD об установленной задаче дает возможность быстрой активизации, так как в ней содержатся все параметры, необходимые системе для активизации запрошенной задачи. Число установленных бездействующих задач может значительно превышать число активных задач.

Если раздел, в котором задача установлена, занят, то она помещается в очередь активных задач и в соответствии со своим приоритетом может конкурировать за системные ресурсы (за требуемый раздел).

Приоритет задачи определяется числом, назначенным задаче при ее построении, установке или при выполнении. Приоритет находится в пределах от 1 до 250 (десятичное значение), причем большее число соответствует более высокому приоритету. Если текущая задача блокирована (например, ждет завершения ввода-вывода), управляющая задача выбирает из очереди активных задач задачу с наивысшим приоритетом и предоставляет ей процессор.

В ОСРВ выполнение задач реального времени совмещается с менее срочными работами, причем задачи реального времени имеют более высокий приоритет. Такое соглашение принято для того, чтобы управляющая программа отдавала предпочтение задачам реального време-

ни. При распределении оперативной памяти управляющая программа может освободить раздел или подраздел для более приоритетной задачи путем *выгрузки на диск* менее приоритетной задачи. Выгруженная задача будет конкурировать в дальнейшем за ресурсы системы в соответствии со своим приоритетом. С целью обеспечения требуемого времени реакции в системах реального времени отдельные задачи объявляются невыгружаемыми. В этом случае даже блокированная задача не может быть выгружена на диск.

Для выгрузки задачи на диске должно быть достаточное пространство выгрузки, равное размеру задачи (или размеру раздела, в котором выполнялась задача). Пространство выгрузки распределяется либо *статически* при построении задачи, либо *динамически* во время выполнения. В ОСРВ можно использовать любой из этих способов выделения пространства выгрузки.

При построении задачи пользователь может затребовать, чтобы пространство выгрузки было размещено в файле образа задачи, который содержит сегменты задачи. В этом случае при выполнении задачи на диске всегда имеется пространство выгрузки для задачи, независимо от того, будет ли в действительности управляющая программа выгружать эту задачу.

Динамическое выделение пространства выгрузки позволяет более эффективно использовать диск. Вместо выделения на диске пространства, равного суммарному размеру всех выгружаемых задач, можно создать один файл выгрузки, динамически выделяя и освобождая пространство в файле при выгрузке и загрузке задач. Этот файл создается по команде с терминала. Недостатком динамического распределения пространства выгрузки является то, что в файле выгрузки может не оказаться свободного пространства для выгружаемой задачи.

Управляющая программа распределяет время процессора и память на основе приоритетов задач. В системе ОСРВ применяются различные режимы диспетчеризации. Два из них: *круговая диспетчеризация* и *свопинг* — влияют на распределение ресурсов, когда многие активные задачи имеют равные приоритеты.

Если несколько находящихся в памяти задач имеют одинаковые приоритеты, управляющая программа стремится отдать время процессора тем задачам, которые оказываются первыми в таблице STD (записи в STD для задач с равными приоритетами обычно появляются в порядке, в котором задачи были установлены). Чтобы

сделать такие задачи равноправными, в ОСРВ используется *круговая диспетчеризация,* при которой периодически просматриваются активные задачи и на каждом уровне приоритетов выполняется циклическое продвижение задач к началу очереди.

Иная проблема возникает, когда несколько активных задач с равными приоритетами конкурируют за раздел. Обычно задача не может заставить управляющую программу выгрузить из раздела другую задачу с тем же или более высоким приоритетом. *Свопинг* позволяет задачам с равными приоритетами выгружать друг друга из раздела. Когда задача начинает выполняться, управляющая программа добавляет приоритет свопинга к приоритету выполнения задачи. Во время выполнения задачи управляющая программа постоянно уменьшает приоритет свопинга, который в итоге принимает отрицательное значение. Если сумма уменьшенного приоритета свопинга и приоритета, с которым выполняется задача, меньше, чем приоритет конкурирующей за раздел задачи, то управляющая программа выгружает выполняемую задачу, чтобы отдать пространство раздела конкурирующей задаче.

Управляющая программа помещает выгруженную задачу в конец очереди активных задач, конкурирующих за память на этом уровне приоритета. Приоритет свопинга не влияет ни на распределение времени процессора, ни на диспетчеризацию ввода-вывода.

Некоторые ситуации, называемые *важными событиями,* заставляют управляющую программу просматривать очередь активных задач и выбирать готовую к выполнению задачу с высшим приоритетом. К важным событиям относятся: завершение ввода-вывода, завершение задачи, изменение порядка очереди задач, выполнение одной или ряда системных директив, выданных задачей, истечение интервала времени при круговой диспетчеризации.

1.3.3. ФУНКЦИИ СИСТЕМНЫХ ДИРЕКТИВ

Системная директива — это запрос задачи к управляющей программе на выполнение требуемой системной функции. Пользовательские задачи используют системные директивы для управления вводом-выводом, выполнением и взаимодействием задач. Большинство важных событий связано прямо или косвенно с выполнением системных директив. Системные директивы влияют на то, каким образом

управляющая программа распределяет системные ресурсы между активными задачами.

Системные директивы позволяют задачам выполнять следующие функции:

получать информацию о системе и о задаче;

измерять временной интервал;

выполнять операцию ввода-вывода;

выполнять действия с логическим или виртуальным адресным пространством задачи;

приостанавливать и возобновлять выполнение задачи;

запросить запуск другой задачи;

успешно или аварийно завершать выполнение задачи;

породить одну задачу из другой, обеспечить взаимосвязь между ними, получить информацию об обеих задачах.

Системные директивы позволяют задачам использовать некоторые основные средства системы: флаги событий, системные прерывания и расширенное логическое пространство.

Ожидание некоторого события в системе и в задачах может быть связано со значением *флага события*. Задачи могут использовать флаги событий для синхронизации своей работы и взаимодействия с другими задачами. Когда происходит важное событие, управляющая программа устанавливает флаг события. Задача может опрашивать состояние флага и тем самым определять, произошло ли соответствующее событие. Задача может и сама выдать системную директиву установки флага.

В ОСРВ имеется 96 флагов событий. Каждый флаг события имеет соответствующий номер. Первые 32 флага являются *локальными* для каждой задачи и устанавливаются или сбрасываются в результате действий самой задачи. Следующие 32 флага являются общими для всех задач и называются *общими флагами*. Общие флаги могут быть установлены или сброшены в результате действий любой задачи. Последние восемь флагов в каждой группе 25—32 — локальные флаги и общие флаги 57—64 зарезервированы для системы. Флаги третьей группы (65—96) называются *групповыми глобальными флагами*. Эти флаги используются в тех же случаях, что и общие флаги, но ими могут пользоваться лишь задачи, выполняющиеся под кодом идентификации с номером группы, для которой эти флаги созданы.

1.3.4. ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

В системе реального времени гибкость, быстродействие и разнообразие средств реакции на внешние события имеют решающее значение. Полностью возможности быстродействия оборудования могут использоваться программой, выполняющейся на машине без операционной системы. В этом случае программа сама должна обеспечивать реакции на синхронные и асинхронные прерывания.

Синхронные прерывания вызываются самой программой (например, резервная инструкция, EMT и др.) в отличие от *асинхронных,* вызванных внешними причинами по отношению к выполнению программы.

Системные средства обслуживания прерываний позволяют передать управление для обработки прерывания в указанную точку задачи пользователя, когда происходит определенное событие. Они позволяют задачам реагировать на синхронные и асинхронные события. В ОСРВ различают синхронные системные прерывания SST и асинхронные системные прерывания AST.

Задача может иметь программу реакции на все виды синхронных прерываний процессора за исключением инструкции EMT с кодом 377, зарезервированной для обращений к системе.

В ОСРВ имеется широкий набор возможностей обработки асинхронных прерываний. Прерывание может быть обслужено средствами системы или задачей пользователя. В первом случае прерывание обслуживается драйвером устройства или задачей, выполнившей директиву CINT⊙. Прерывание может обслуживаться на высшем, седьмом, приоритете процессора (т. е. в режиме запрета других прерываний), на приоритете устройства или на нулевом приоритете процессора. В ОСРВ пользователям предоставляется возможность разработки и включения в систему собственных драйверов и привилегированных задач, которые обращаются непосредственно к регистрам устройств.

При обработке прерываний задачей пользователя имеется возможность выбора режима обслуживания прерываний. Задача может быть фиксирована в оперативной памяти, при этом она конкурирует за процессор в соответствии со своим приоритетом. Нефиксированная в памяти задача может быть установлена в разделе, управляемом пользователем или системой. Все эти детали влияют на время реакции задачи на конкретное событие. Кроме того, время реакции будет зависеть от динамической ситуации в разделе.

2

ДИРЕКТИВЫ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

2.1. ВВЕДЕНИЕ В УПРАВЛЯЮЩУЮ ПРОГРАММУ

Ядром системы ОСРВ (п. 1.2.1) является управляющая программа, которая организует доступ задач к ресурсам, поддерживает соответствие между виртуальными и физическими адресными пространствами, исполняет программные запросы задач к системе (см. также п. 1.3).

Все программные вызовы, называемые *директивами управляющей программы,* реализуются выполнением инструкции EMT 377.

Данные, указанные в директиве, передаются системе через блок параметров директивы DPB, младший байт которого содержит код идентификации директивы (нечетное число), следующий за ним байт содержит количество слов в блоке, остальные байты содержат параметры, относящиеся к данной директиве. В результате обработки директивы система формирует *слово состояния директивы* DSW со значением +1 при успешном завершении директивы и отрицательным числом (код аварийного завершения директивы), если директива отвергнута. Значение слова состояния директивы может быть прочитано из слова с именем ⊙DSW.

Дополнительно директива (кроме директив, завершающих задачи) возвращает управление следующей за ней инструкции с установкой кода условия C: 0 — если директива выполнена успешно или 1 — если директива отвергнута.

2.2. ОБЩИЕ СОГЛАШЕНИЯ ПО ОПИСАНИЮ ФОРМАТОВ МАКРОВЫЗОВОВ И ВЫЗОВОВ ПОДПРОГРАММ

Вызов директивы в программе, написанной на языке макроассемблера, задается в виде макровызовов [4]. Соответствующие директивам макроопределения содержатся в системной макробиблиотеке LB: [1,1] OCMAC.SML.

Используемые в программе макроопределения предварительно должны быть вызваны директивой .MCALL. Большинство директив могут иметь три формы вызова.

Имя макроопределения директивы содержит не более четырех символов, за которыми следует знак ⊙ и еще один необязательный символ, кодирующий форму директивы.

Если необязательный символ отсутствует (⊙ — форма директивы), формируется только блок параметров директивы DPB, который размещается в точке макровызова и не содержит машинных инструкций. Эта форма обычно используется в сочетании с макрокомандой DIR⊙ (см. ниже).

Если необязательным символом служит буква S (⊙S-форма), формируются инструкции засылки блока DPB в стек и инструкция EMT 377. Эта форма в отличие от предыдущей обеспечивает повторную входимость программ.

Если необязательным параметром служит буква C (⊙C-форма), то адрес блока DPB засылается в стек и выполняется инструкция EMT 377. Блок DPB создается в отдельной программной секции с именем ⊙DPB⊙⊙. В исходной программной секции генерируется команда засылки адреса DPB в стек и EMT 377.

Чтобы обеспечить возврат в исходную программную секцию, пользователь должен указать имя этой секции в директиве непосредственно после параметров самой директивы. Если имя секции не указано, предполагается неименованная секция.

Макрокоманда DIR⊙A порождает расширение:

```
MOV A, — (SP)
EMT 377
```

где A — адрес блока DPB.

Директивы в ⊙S- и ⊙C-формах, а также макрокоманда DIR⊙ допускают необязательный последний аргумент, который указывает адрес пользовательской про-

граммы обработки ошибок директивы. Если он задан (B), то генерируются инструкции

```
BCC.+n
JSR PC,B
```

где n — смещение в словах до инструкции, следующей после этих двух.

Большинство макроопределений в ⊙- и ⊙C-формах порождают символы, определяющие величины смещений в байтах от начала блока до соответствующего его элемента.

Если в программе определен символ ⊙⊙⊙GBL (⊙⊙⊙GBL=0), эти формы не порождают блока DPB, а генерируют глобальные символы смещений.

При программировании задач на языке Фортран используется набор подпрограмм, выполняющих функции вызова системных директив. Эти подпрограммы содержатся в библиотеке объектных модулей, используемой при построении задачи, написанной на языке Фортран. Для обращения к любой из них программист должен использовать в программе, написанной на Фортране, оператор CALL с именем соответствующей подпрограммы. Для передачи в подпрограмму необходимых аргументов должны соблюдаться следующие правила.

Аргумент (имя задачи) определяется переменной типа REAL и кодируется в коде RADIX-50. Эта переменная может быть определена на этапе компиляции оператором DATA.

Аргумент (целый массив), содержащий адреса переменных и массивов, заполняется посредством вызова подпрограммы GETADR. Вызов CALL GETADR (M,[A1], [A2],...,[An]) позволяет записать адреса аргументов A1 и т. д. в последовательные элементы массива M. Если аргумент опущен, запись в соответствующий элемент массива не делается.

Каждый вызов директивы на Фортране может включать необязательный аргумент — код завершения директивы (кдз). Его значения соответствуют значениям кода завершения директивы, возвращаемым в слове состояния директивы ⊙DSW. При выполнении директив управляющая программа сохраняет значения общих регистров процессора*.

---

* Пробелы между запятыми и параметрами директив, а также между символами] [не являются элементами директив, а вставлены для удобства чтения текста.

2.3. ГРУППЫ ДИРЕКТИВ

Директивы группируются по следующим функциям:
директивы управления выполнением задач;
директивы управления состоянием задач;
информационные директивы;
директивы, связанные с событиями;
директивы, связанные с обработкой прерываний;
директивы ввода-вывода и межзадачных связей;
директивы управления памятью;
директивы порождения задач;
директивы, предназначенные для интерпретаторов командных строк CLI.

Имеются ограничения на использование некоторых директив.

ABRT⊙ — аварийно завершить задачу — в системах с многопользовательской защитой задача, запущенная с непривилегированного терминала, может завершить аварийно лишь задачи, имеющие в качестве устройства TI: тот же терминал, что и сама задача;

ALTP⊙ — изменить приоритет задачи — в системах с многопользовательской защитой задача, запущенная с непривилегированного терминала, может изменить свой приоритет на значение меньше приоритета или равное приоритету, заданному при установке задачи;

CSRQ⊙ — отменить требования запуска задачи по времени — в системах с многопользовательской защитой задача, запущенная с непривилегированного терминала, может издавать директиву только для задач, имеющих в качестве устройства TI: тот же терминал, что и сама задача;

CINT⊙ — подключиться к вектору прерывания — в системе с диспетчером памяти непривилегированная задача не может использовать эту директиву;

SCLI⊙ — назначить интерпретатор командных строк CLI терминалу — в некоторых случаях эта директива не может быть использована непривилегированной задачей.

Директивы управляющей программы рассматриваются по группам в алфавитном порядке. При описании аргументов директив используются следующие основные обозначения:

имя — имя задачи;

кдз — переменная типа I×2 для кода завершения директивы (при вызове на языке Фортран);

ош — адрес программы обработки ошибок;

пар — имя четырехсловного целого массива, пар(1) — первая половина имени раздела (в RADIX-50), пар(2) — вторая поло-

вина имени раздела (в RADIX-50), пар(3) — приоритет, пар(4) — код идентификации пользователя;

рзд — имя раздела;

прт — приоритет;

гр — номер группы в коде идентификации пользователя;

чл — номер члена в коде идентификации пользователя;

прщ — приращение интервала времени от момента издания директивы до начала запроса запуска задачи;

едн — единица измерения интервала;

инт — интервал передиспетчеризации;

едн1 — единица измерения интервала передиспетчеризации;

идр — идентификатор района;

прклч — целая переменная для значения консольных переключателей;

флг — номер флага события;

аст — адрес подпрограммы обработки асинхронного системного прерывания (AST-программы);

нгр — номер группы флагов;

врем — величина интервала времени;

мск — 16-битная маска флагов;

прм — имя целого массива для получения параметров (адрес параметров для макровызова);

чпрм — число параметров, возвращаемых в массив параметров;

пуст — параметр резервируется для дальнейшего развития системы;

тбл — адрес таблицы точек входа (SST-вектор) подпрограмм обработки синхронных системных прерываний (SST-программ);

дл — длина (число элементов) таблицы в словах;

лгн — логический номер устройства;

иустр — имя устройства (формат: А2 на Фортране);

нустр — номер устройства;

век — адрес вектора прерываний. Должен быть задан в границах от 60 (восьмеричное) до максимального адреса вектора, определенного при генерации, и должен быть кратен четырем;

баз — виртуальный базовый адрес для отображения программы обработки прерываний и программы разделения (запрещения прерываний при помощи системной пары регистров диспетчера памяти APR5;

адрпр — виртуальный адрес программы обработки прерываний или 0 для отключения задачи от вектора прерываний;

рзадр — виртуальный адрес программы разрешения (запрещения) прерываний на устройстве;

кдф — код функции;

ствв — имя двухсловного целого массива для засылки информации о завершаемой операции ввода-вывода;

буф — имя целого массива (адрес буфера для макровызова), содержащего данные;

длбуф — имя целой переменной, содержащей длину буфера (длина буфера в байтах для макровызова);

прм # 1,..., прм # N — список параметров, зависящих от задачи, для которой посылается сообщение;

бор — имя 8-словного целого массива, содержащего RDB — блок описания района (адрес блока описания района для макровызова);

боо — имя 8-словного целого массива, содержащего WDB — блок описания окна (адрес WDB для макровызова);

тбл — имя целого массива (адрес таблицы для макровызова) для получения описания существующих в текущий момент адресных окон;

дпб — имя 10-словного целого массива, используемого как дополнительный буфер данных;

инф — имя 8-словного целого массива, содержащего дополнительную информацию;

блс — имя 8-словного блока состояния, который заполняется при завершении задачи-потомка или при выдаче его состояния (адрес блока состояния для макровызова);

сост — 16-битная величина, передаваемая задаче-родителю;

имяр — имя массива (или целая переменная формата 1×4), содержащего имя (в RADIX-50) задачи-родителя, передаваемого запрашиваемой задаче;

флб — флажковый байт, задающий функции директивы;

стр — адрес командной строки для порождаемой задачи;

длстр — длина командной строки (максимальная длина — 79 байт);

ибуф — имя целого массива (информационного буфера), куда передается вспомогательная информация для CLI;

длнбуф — имя целой переменной, содержащей длину информационного буфера;

адрком — имя целой переменной, содержащей адрес системного буфера, в котором лежит команда CLI (адрес команды в системном буфере для макровызова).

2.3.1. ДИРЕКТИВЫ УПРАВЛЕНИЯ ВЫПОЛНЕНИЕМ ЗАДАЧ

Директивы управления выполнением задач обеспечивают главным образом запуск и останов задач. Каждая из этих директив (кроме директивы расширения памяти задачи) приводит к изменению состояния задачи, если задача еще не находится в запрашиваемом состоянии.

Директива ABRT⊙ — аварийно завершить задачу — завершает выполнение указанной задачи. Вызов на языке Фортран:

```
CALL ABORT (имя[,кдз])
```

Макровызов:

```
ABRT⊙ имя
```

Директива CSRQ⊙ — отменить требования запуска задачи по времени — отменяет все требования на запуск по времени указанной задачи, независимо от источников требований — являются ли эти требования результатом директивы RUN⊙ (запустить задачу в указанное время) либо результатом одной из форм команды RUN программы связи с оператором. Вызов на языке Фортран:

```
CALL CANALL (имя,кдз)
```

Макровызов:

```
CSRQ⊙ имя
```

Директива EXIT⊙S —завершить задачу — завершает задачу, издавшую директиву. Рекомендуется ⊙S-форма. Вызов на языке Фортран:

```
STOP [сооб]
```

или

```
CALL EXIT
```

где сооб — строка, содержащая от 1 до 26 символов; строка выводится на терминал при выполнении STOP

Макровызов:

```
EXIT⊙S ош
```

Директива EXTK⊙ — расширить память задачи — изменяет размер памяти задачи, издавшей директиву, на положительную или отрицательную величину, заданную в 32-словных блоках. Вызов на языке Фортран:

```
CALL EXTTSK ([прир] [,кдз])
```

Макровызов:

```
EXTK⊙ [прир]
```

где прир — положительное или отрицательное число, равное числу 32-словных блоков, на которое увеличивается или уменьшается размер задачи.

Директива RQST⊙ — запустить задачу — активизирует задачу. Активная задача будет запущена на выполнение в соответствии с приоритетом при условии выделения ей необходимой памяти. RQST⊙ является основным механизмом, используемым выполняющимися задачами для инициации других, установленных в системе, но бездействующих задач. Вызов на языке Фортран:

```
CALL REQUES (имя,[,пар] [,кдз])
```

Макровызов:

```
RQST⊙ имя,[,рзд] ,[прт] [,гр,чл]
```

Имя раздела и приоритет игнорируются, но должны присутствовать.

Директива RSUM⊙ — возобновить задачу — возобновляет выполнение задачи, издавшей директиву SPND⊙ (приостановить задачу). Вызов на языке Фортран:

```
CALL RESUME (имя [,кдз])
```

Макровызов:

```
RSUM⊙ имя
```

Директива SPND⊙ — приостановить задачу — приостанавливает выполнение задачи, издавшей директиву. Рекомендуется ⊙S-форма. Задача может приостановить только себя и может быть запущена снова директивой RSUM⊙ — возобновить задачу или командой оператора RES. Вызов на языке Фортран:

CALL SUSPEND [(кдз)]

Макровызов:

SPND⊙S [ош]

Директива RUN⊙ — запустить задачу в указанное время — заставляет систему активизировать задачу в определенный момент времени и по желанию пользователя периодически повторять запуск этой задачи.

Время запуска задачи задается при помощи приращения относительно момента издания директивы. Если параметры прщ, инт, едн1 опущены, то директива RUN⊙ подобна директиве RQST⊙ — (запустить задачу) с тем отличием, что, во-первых, задача будет активизирована через одну единицу таймера (тик) от момента издания директивы и, во-вторых, система всегда устанавливает в качестве устройства TI: запрошенной задачи устройство CO: Вызов на языке Фортран:

CALL RUN (имя, [пар] , [прщ] , едн , [инт] , [едн1] [,кдз])

или

CALL START (имя, прщ, едн [,кдз])

Макровызов:

RUN⊙ имя, [рзд] ,[прт] , [гр] , [чл] , [прщ] , едн [, инт, едн1]

2.3.2. ДИРЕКТИВЫ, УПРАВЛЯЮЩИЕ СОСТОЯНИЕМ ЗАДАЧ

Первая директива этой группы изменяет приоритет выполнения задачи, а две другие влияют на свойство выгружаемости.

Директива ALTP⊙ — изменить приоритет задачи — изменяет приоритет выполнения активной задачи, указанной в этой директиве, следующим образом: устанавливает приоритет, указанный в директиве, а если его значение не задано, устанавливает приоритет задачи, равный значению приоритета, полученному задачей при установке. Вызов на языке Фортран:

CALL ALTPRI ([имя] , [прт] [,кдз])

Макровызов:

ALTP⊙ [имя] [,прт]

Директива DSCP⊙S — запретить выгружаемость задачи — запрещает выгружаемость задачи, которая была установлена как выгружаемая. Директива DSCP⊙ воздействует только на задачу, издавшую директиву. Не может быть запрещена выгружаемость другой задачи. Вызов на языке Фортран:

CALL DISCKP [(кдз)]

Макровызов:

DSCP⊙S [ош]

Рекомендуется ⊙S-форма директивы.

Директива ENCP⊙S — разрешить выгружаемость задачи — возобновляет выгружаемость задачи, установленной как выгружаемая, т. е. прекращает действие директивы DSCP⊙S — запретить выгружаемость задачи. Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран:

CALL ENACKP [(кдз)]

Макровызов:

ENCP⊙S [ош]

2.3.3. ИНФОРМАЦИОННЫЕ ДИРЕКТИВЫ

Пять информационных директив передают запрашивающей задаче данные, сохраняемые в системе: параметры задачи, состояние консольных переключателей, параметры раздела или района, время дня.

Директива GPRT⊙ — дать параметры раздела — предоставляет информацию о разделе в трехсловном буфере. Если раздел не указан, то предполагается раздел задачи, издавшей директиву. Вызов на языке Фортран:

CALL GETPAR ([рзд] , буф [,кдз])

где буф — трехсловный целый массив для параметров раздела.

Макровызов:

GPRT⊙ [рзд] , буф

где буф — адрес трехсловного буфера для параметров раздела.

Формат буфера:

буф (0) — базовый физический адрес раздела в 32-словных блоках

(раздел всегда выравнен на границу 32-словного блока). Так, для раздела, начинающегося с 40000_8, это слово будет содержать 400_8;

буф (1) — размер раздела в 32-словных блоках;

буф (2) — слово флагов раздела; это слово устанавливается в 0, если раздел системно-управляемый, и в 1, если это пользовательский раздел.

Директива GREG⊙ — дать параметры района — записывает параметры района в трехсловный буфер. Если район не задан, то подразумевается собственный район задачи.

Вызов на языке Фортран:

```
CALL GETREG ([идр] , буф[,кдз] )
```

где буф — трехсловный целый массив для параметров района.

Макровызов:

```
GREG⊙ [идр] , буф
```

где буф — адрес трехсловного буфера для параметров района.

Формат буфера:

буф (0) — базовый адрес района в 32-словных блоках (район всегда выравнен на границу 32-словного блока). Например, начало района 1000_8 передается как 10_8;

буф (1) — размер района в 32-словных блоках;

буф (2) — тип района; слово равно 0, если район размещен в системно-управляемом разделе, и равно 1 для района, размещенного в разделе, управляемом пользователем.

Директива GSSW⊙S — дать значение консольных переключателей — записывает содержимое регистра переключателей процессора в слово состояния директивы задачи. Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран:

```
CALL READSW (прклч)
```

Макровызов:

```
GSSW⊙S [ош]
```

Директива GTIM⊙ — дать параметры времени — заполняет указанный 8-словный буфер значением текущего времени. Все параметры времени возвращаются, как двоичные числа. Десятичный диапазон величин приведен ниже в описании формата буфера. Вызов на языке Фортран:

```
CALL GETTIM (буф [,кдз] )
```

где буф — имя целого 8-словного массива для получения параметров времени.

Макровызов:

GTIM⊙ буф

где буф — адрес 8-словного буфера.

Формат буфера:

буф (0) — год (начиная с 1900);
буф (1) — месяц года (1—12);
буф (2) — день месяца (1—31);
буф (3) — час дня (0—23);
буф (4) — минуты (0—59);
буф (5) — секунды (0—59);
буф (6) — тики (0—49);
буф (7) — число тиков в секунду (50).

Директива GTSK⊙ — дать параметры задачи — заполняет указанный 16-словный буфер параметрами задачи, издавшей директиву. Вызов на языке Фортран:

CALL GETTSK (буф [,кдз])

где буф — имя 16-словного целого массива для получения параметров задачи.

Макровызов:

GTSK⊙ буф

где буф — адрес 16-словного буфера.

Формат буфера:

буф (1) — имя задачи, издавшей директиву (первая половина имени в RAD-50);
буф (2) — имя задачи, издавшей директиву (вторая половина имени в RAD-50);
буф (3) — имя раздела (первая половина имени в RAD-50),
буф (4) — имя раздела (вторая половина имени в RAD-50),
буф (5) — резерв;
буф (6) — резерв;
буф (7) — приоритет выполнения;
буф (8) — код идентификации пользователя для задачи, издавшей директиву; в системе с многопользовательской защитой — код UIC задачи по умолчанию;
буф (9) — число логических номеров устройств ввода-вывода (LUN);
буф (10) — резерв;
буф (11) — резерв;
буф (12) — адрес таблицы SST-векторов задачи;
буф (13) — размер таблицы SST-векторов задачи в словах;
буф (14) — в системе с диспетчером памяти размер нулевого адресного окна задачи (в байтах); в системах без диспетчера памяти размер раздела задачи (в байтах);
буф (15) — признак версии системы, в которой выполняется задача;
буф (16) — код UIC по защите (в системе с многопользовательской защитой — UIC регистрации пользователя)

2.3.4 ДИРЕКТИВЫ, СВЯЗАННЫЕ С СОБЫТИЯМИ

Директивы, связанные с событиями и флагами событий, являются средством, предназначенным для синхронизации между задачами и внутри задач. Директивы этой группы позволяют также устанавливать системное время. Необходима осторожность в использовании этих директив, так как ошибки, возникающие в результате неверной синхронизации, трудно устранить.

Директива CLEF⊙ очистить флаг — обнуляет указанный флаг события и сообщает кодом завершения директивы значение флага перед его очисткой. Вызов на языке Фортран:

```
CALL CLREF (флг [,кдз])
```

Макровызов:

```
CLEF⊙ флг
```

Директива CMKT — отменить запросы директивы MRKT⊙ — отменяет либо указанные запросы директивы MRKT⊙, либо все запросы директивы MRKT⊙, изданные данной задачей. Вызов на языке Фортран:

```
CALL CANMT([флг] [,кдз])
```

Макровызов:

```
CMKT⊙ [флг,аст,ош]
```

Если в директиве не заданы ни флг, ни аст, то директива отменяет все запросы директивы MRKT⊙, изданные данной задачей.

Директива CRGF⊙ — создать групповые глобальные флаги — создает блок описания групповых глобальных флагов GFB и включает его в список имеющихся GFB. Если в момент издания директивы блока GFB для указанной группы пользователей не существует, то такой блок создается, все флаги в нем обнуляются. Если же в момент издания директивы блок GFB для указанной группы существует, то новый блок GFB не создается, флаги в имеющемся GFB не изменяются. Если блок GFB отмечен к удалению по директиве ELGF⊙, то по директиве CRGF⊙ в нем очищается бит удаления GS.DEL. Вызов на языке Фортран:

```
CALL CRGF ([нгр] [,кдз])
```

Макровызов:

```
CRGF⊙ [нгр]
```

Директива DECL⊙S — объявить важное событие — позволяет управляющей программе осуществлять передиспетчеризацию задач. Эту директиву нужно использовать осторожно, так как ее беспорядочное применение приводит к чрезмерному возрастанию накладных расходов системы. Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран:

```
CALL DECLAR ([,кдз])
```

Макровызов:

```
DECL⊙S [,ош]
```

Директива ELGF⊙ — уничтожить группу глобальных флагов — отмечает групповые глобальные флаги к уничтожению. Если данные групповые глобальные флаги не ожидает ни одна задача (счетчик доступа к флагам G.CNT в блоке GFB равен нулю), то блок GFB уничтожается (исключается из списка блоков GFB), и память, занятая под блок, освобождается. Если есть задачи, использующие данные групповые глобальные флаги, то управляющая программа только отмечает блок к уничтожению — устанавливает бит GS.DEL. Блок GFB, отмеченный к удалению, будет уничтожен после того, как не останется задач, ожидающих эти флаги. Однако, если до того, как блок GFB был уничтожен, издана директива CRGF⊙, то бит GS.DEL очищается. Вызов на языке Фортран:

```
CALL ELGF ([нгр] [,кдз])
```

Макровызов:

```
ELGF⊙ [нгр]
```

Директива EXIF⊙ — завершить задачу, если не установлен флаг — завершает задачу, издавшую директиву, в том случае, если указанный флаг события не установлен. Если флаг установлен, то управляющая программа возвращает управление задаче. Вызов на языке Фортран:

```
CALL EXITIF (флг [,кдз])
```

Макровызов:

```
EXIF⊙ флг
```

Директива MRKT⊙ — объявить важное событие через временной интервал. По этой директиве через указанный интервал времени происходит важное событие. Интервал отсчитывается с момента издания директивы. Если указан флаг события, то при издании директивы он обнуляется и устанавливается в момент объявления важного

события. Если указан адрес входа в AST-программу, то в момент объявления важного события для задачи объявляется асинхронное системное прерывание, которое будет обрабатываться этой AST-программой. На входе в AST-программу в стек будет записано PS, PC задачи в точке прерывания, состояние директивы, слово флагов, ожидаемых задачей, номер флага, указанного в директиве. Если ни номер флага события, ни адрес программы обработки AST не заданы, важное событие все равно будет объявлено по истечении указанного интервала времени. Вызов на языке Фортран:

```
CALL MARK (флг , врем , едн [,кдз])
```

Макровызов:

```
MRKT⊙ [флг] , врем , едн [,ACT]
```

Директива RDAF⊙ — считать все флаги — считывает 64 флага события для задачи, издавшей директиву, и записывает их значение в 64-битный (4-словный) буфер. Эта директива не информирует о значении групповых глобальных флагов (флагов 65—96). Вызов на языке Фортран:

```
CALL READEF (флг [,кдз])
```

Подпрограмма READEF считывает только один указанный флаг и возвращает значение в кдз.

Макровызов:

```
RDAF⊙ буф
```

где буф — адрес 4-словного буфера.

Формат буфера:

буф (0) — локальные флаги задачи 1—16;
буф (1) — локальные флаги задачи 17—32;
буф (2) — общие флаги 33—38;
буф (3) — общие флаги 49—64.

Директива RDXF⊙ — считать флаги — считывает для задачи, выполнившей директиву, все локальные, общие и групповые глобальные флаги и записывает в 96-битный (6-словный) буфер их значения. Вызов на языке Фортран:

```
CALL READEF(флг [,кдз])
```

Подпрограмма READEF может считать для задачи только один флаг.

Макровызов:

```
RDXF⊙ буф
```

Формат буфера:

буф (0) — локальные флаги задачи 1—16;
буф (1) — локальные флаги задачи 17—32;
буф (2) — общие флаги 33—48;
буф (3) — общие флаги 49—64;
буф (4) — групповые глобальные флаги задачи 65—80;
буф (5) — групповые глобальные флаги задачи 81—96.

Директива SETF⊙ — установить флаг — устанавливает указанный флаг события и передает в слове состояния директивы значение флага, которое было перед его установкой. Вызов на языке Фортран:

```
CALL SETEF (флг [,кдз])
```

Макровызов:

```
SETF⊙ флг
```

Директива STLO⊙ — остановить задачу до установки дизъюнкции флагов — останавливает задачу, издавшую директиву, на время, пока не будет установлен один или несколько указанных флагов из группы флагов. Возможны следующие группы флагов:

группа 0 — локальные флаги 1—16;
группа 1 — локальные флаги 17—32;
группа 2 — общие флаги 33—48;
группа 3 — общие флаги 49—64;
группа 4 — групповые глобальные флаги 65—80;
группа 5 — групповые глобальные флаги 81—96.

Если в момент издания директивы какие-либо флаги из числа указанных уже установлены, то задача не будет прервана. Директива STLO⊙ не может быть издана из AST-программы. Задача, остановленная до установки дизъюнкции флагов, может быть выведена из останова только за счет установки одного или нескольких указанных флагов; вывести из останова такую задачу с помощью директивы USTP (вывести задачу из останова) или команды оператора UNSTOP нельзя. Вызов на языке Фортран:

```
CALL STLOR (флг1 , флг2 , ... , флг N)
```

Макровызов:

```
STLO⊙ нгр , мск
```

Директива STOP⊙ — остановить задачу — останавливает задачу, выполнившую директиву. Директива STOP⊙ не может быть выполнена задачей в момент обслуживания AST-прерывания, т. е. находящейся в AST-состоянии. Задача, остановленная по STOP⊙S, может быть активизирована следующим образом:

по директиве USTP⊙, выполненной другой задачей;

по директиве USTP⊙, выполненной данной задачей в AST-программе;

по команде оператора UNSTOP.

Рекомендуется ⊙S-форма макровызова. Вызов на языке Фортран:

```
CALL STOP ([кдз])
```

Макровызов:

```
STOP⊙S
```

Директива STSE⊙ — остановить задачу до установки флага — останавливает задачу, выполнившую директиву, до тех пор, пока не будет установлен указанный флаг. Если в момент выполнения директивы флаг уже установлен, то задача не будет остановлена. Эта директива не может выполняться в AST-программе.

Задача, остановленная по директиве STSE⊙, так же, как и в случае директивы STLO⊙, может быть выведена из останова только при установке соответствующего флага; директива USTP⊙ или команда оператора UNSTOP не могут вывести такую задачу из останова. Вызов на языке Фортран:

```
CALL STOPFR (флг [,кдз])
```

Макровызов:

```
STSE⊙ флг
```

Директива ULGF⊙ — разблокировать групповые глобальные флаги — уменьшает на единицу счетчик использования групповых глобальных флагов для группы, указанной в коде UIC защиты задачи (поле H.CUIC+1 в заголовке задачи). Директива разблокирует групповые флаги, которые блокируются при создании по директиве CRGF⊙ (создать групповые глобальные флаги). Задача должна разблокировать флаги, прежде чем блокировать их снова. Групповые глобальные флаги уничтожаются, если выполняются следующие два условия:

счетчик доступа к флагам в блоке GFB в результате выполнения директивы ULGF⊙ обнулен;

блок GFB отмечен на удаление.

Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран:

```
CALL ULGF [(кдз)]
```

Макровызов:

```
ULGF⊙S [ош]
```

Директива USTP⊙ — вывести задачу из останова — выводит из останова задачу, которая остановила себя директивой STOP⊙ или RCST⊙. Эта директива не применима к задаче, остановленной до установки одного флага или дизъюнкции флагов, либо остановленной на время буферированного ввода-вывода. Если задача остановлена по директиве STOP⊙ или RCST⊙, но в момент выполнения директивы USTP⊙ обрабатывала асинхронное прерывание, то останов задачи будет снят только после завершения обработки. Директива USTP⊙ может быть выполнена задачей, запущенной с привилегированного или непривилегированного терминала. Директива USTP⊙ не вызывает объявления важного события. Вызов на языке Фортран:

CALL USTP (имя [,кдз])

Макровызов:

USTP⊙ имя

Директива WSIG⊙S — ждать важное событие — используется, чтобы приостановить решение задачи, выполнившей директиву, до тех пор, пока в системе не произойдет важное событие. Это средство особенно эффективно для приостанова задачи, которая не может продолжаться из-за недостатка динамической памяти, так как важные события, имеющие место в системе, часто приводят к освобождению динамической памяти. Выполнение WSIG⊙S не порождает важного события. Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран:

CALL WFSNE

Макровызов:

WSIG⊙S [ош]

Директива WTLO⊙ — ждать дизъюнкции флагов — блокирует выполнение задачи до тех пор, пока не будет установлен один флаг или комбинация любых из указанных флагов событий в любой из групп флагов. Если при издании директивы какой-либо из указанных флагов установлен, то выполнение задачи не приостанавливается. Вызов на языке Фортран:

CALL WFLOR (флг1, флг2, ..., флг N)

Макровызов:

WTLO⊙ нгр, мск

Директива WTSE⊙ — ждать единичный флаг — блокирует выполнение задачи до тех пор, пока не будет установлен указанный флаг события. Если при издании директивы этот флаг события уже установлен, выполнение задачи не приостанавливается. Вызов на языке Фортран:

```
CALL WAITFR (флг [,кдз])
```

Макровызов:

```
WTSE⊙ флг
```

2.3.5 ДИРЕКТИВЫ УПРАВЛЕНИЯ ОБРАБОТКОЙ ПРЕРЫВАНИЙ

Эти директивы обеспечивают возможность использовать систему прерываний для передачи управления в выполняющуюся задачу.

Директива ASTX⊙S — выход из программы обслуживания асинхронного системного прерывания — обеспечивает выход из AST-программы.

Если в очереди AST-прерываний есть требование и обслуживание AST-прерываний не запрещено, то сразу после выхода из AST-программы обслуживается следующее требование. Если очередь требований пуста, то восстанавливается состояние задачи, бывшее до AST-прерывания. Рекомендуется ⊙S-форма директивы. Вызов на языке Фортран отсутствует.

Макровызов:

```
ASTX⊙S [ош]
```

Директива DSAR⊙S или *IHAR⊙S — запретить обслуживание асинхронных системных прерываний* — запрещает немедленное обслуживание асинхронных системных прерываний для задачи, выполнившей директиву. AST-прерывания, которые возникают после выдачи этой директивы, ставятся в очередь и будут обработаны, когда их обслуживание станет возможным. Во время работы AST-программы обслуживание новых прерываний не выполняется. По умолчанию предполагается, что при начальном запуске задачи обслуживание AST-прерываний разрешено. Рекомендуется ⊙S-форма директивы.

Вызов на языке Фортран:

```
CALL DSASTR [(кдз)]
```

или

```
CALL INASTR [(кдз)]
```

Макровызов:

DSAR⊙S [ош]

или

IHAR⊙S [ош]

Директива SCAA⊙ — назначить программу обработки AST-прерываний по получению командной строки — разрешает или запрещает задаче-интерпретатору командных строк CLI выполнять обработку AST-прерываний по получению командной строки с помощью указания адреса AST-программы. Если обработка AST-прерываний по получению командной строки разрешена, то при передаче командной строки CLI-задаче управляющая программа передает управление на соответствующую AST-подпрограмму. Отсутствие адреса AST-программы означает запрет обработки прерываний по получению командной строки. Директива SCAA⊙ может быть использована только задачами — CLI.

При входе в AST-программу стек задачи содержит следующую информацию:

SP+10 — ноль, так как флаги не используются;
SP+6 — PS задачи в точке AST-прерывания;
SP+4 — PS задачи в точке AST-прерывания;
SP+2 — DSW в момент AST-прерывания;
SP+0 — адрес только что переданного задаче командного буфера.

Перед выполнением директивы ASTX⊙ (выход из программы обслуживания AST-прерывания) необходимо удалить из стека адрес командного буфера.

Адресом командного буфера является адрес буфера, указанного в директиве GCCI⊙ (дать команду для CLI). Вызов на языке Фортран отсутствует.

Макровызов:

SCAA⊙ [аст]

Директива SFPA⊙ — назначить программу обработки AST-прерываний по ошибкам команд с плавающей точкой — либо разрешает обработку AST-прерываний по ошибкам команд с плавающей точкой и задает адрес программы обслуживания AST-прерывания этого типа, либо, если адрес AST-программы не указан, извещает систему, что AST-прерывания по ошибкам команд с плавающей точкой для задачи, выполнившей директиву, игнорируются. Вызов на языке Фортран отсутствует.

Макровызов:

SFPA⊙ [аст]

Директива SPRA⊙ — AST-прерывание по восстановлению питания — либо разрешает обработку AST-прерываний по восстановлению питания и дает адрес точки входа AST-программы, либо извещает систему, что для задачи, издавшей эту директиву, AST-прерывание по восстановлению питания игнорируется. Если адрес точки входа в AST-программу задан, то AST-прерывания по восстановлению питания для задачи, издавшей директиву, допустимы и, если такое событие произойдет, управление будет передано AST-программе. Вызов на языке Фортран:

Для разрешения AST этого типа:

EXTERNAL имя

CALL PWRUP (имя)

Для запрещения AST:

CALL PWRUP

Макровызов:

S'PRA⊙ аст

Директива SRDA⊙ — назначить программу обработки AST-прерываний по получению данных — либо разрешает обработку AST-прерываний по получении данных и задает адрес точки входа в AST-программу, либо, если параметр AST не задан, указывает системе, что AST-прерывания по получению данных для задачи, издавшей директиву, больше не обрабатываются. Вызов на языке Фортран отсутствует.

Макровызов:

SRDA ⊙ аст

Директивы SREA⊙ и SREX⊙ — назначить программу обработки AST-прерываний по аварийному завершению задачи — разрешают обработку AST-прерываний, объявляемых по аварийному завершению задачи. Аварийное завершение задачи может произойти по директиве ABRT⊙ или по команде ABO оператора. Директивы SREA⊙ и SREX⊙ позволяют задаче выполнить свою процедуру, подготавливающую задачу к перезапуску в случае ее аварийного завершения. Если адрес AST-программы в директивах не задан, то ранее заданный адрес AST-программы удаляется. Вызов на языке Фортран:

CALL SREA (аст [,кдз])

и

CALL SREX (аст, прм, чпрм, [пуст] [,кдз])

Макровызов:

SREA⊙ [аст]

и

SREX⊙ аст [,пуст]

Директива SRRA⊙ — назначить программу обработки AST-прерываний по получению данных по ссылке — выполняет одну из двух функций:

разрешает AST-прерывания по получению данных, передаваемых по ссылке, для задачи, выполнившей директиву. Она задает адрес AST-программы;

запрещает AST-прерывания по получению данных по ссылке для задачи, выполнившей директиву. В этом случае адрес программы обработки асинхронных прерываний обнуляется. Вызов на языке Фортран отсутствует.

Макровызов:

SRRA⊙ [аст]

Директива SVDB⊙ — задать таблицу SST-векторов для отладочных средств — задает адрес таблицы точек входов SST-программ обработки синхронных системных прерываний для отладочных средств, используемых задачей (например, для отладчика). При отказе от обработки SST-прерываний отладчиком выполняется макровызов без параметров. Если SST-программа определена в таблице SST-вектора, заданной для задачи и в таблице SST-вектора отладочных средств, то прерывание будет обслуживаться программой, заданной в таблице SST-вектора для отладочных средств. Вызов на языке Фортран отсутствует.

Макровызов:

SVDB⊙ [табл] [дл]

Директива SVTK⊙ — задать таблицу SST — вектора для задачи — задает адрес таблицы точек входов программ, обслуживающих синхронные системные прерывания, для задачи, издавшей директиву. Эта таблица называется таблицей SST-вектора. Макровызов без параметров означает отказ от обработки SST-прерываний. Вызов на языке Фортран отсутствует.

Макровызов:

SVTK⊙ [табл] [,дл]

2.3.6 ДИРЕКТИВЫ ВВОДА-ВЫВОДА И МЕЖЗАДАЧНЫХ СВЯЗЕЙ

Директивы ввода-вывода и межзадачных связей обеспечивают задаче доступ к системе ввода-вывода, связь с другими задачами в системе; прием командной строки, посланной задаче через программу связи с оператором при запуске задачи.

Директива ALUN⊙ — назначить логический номер устройства — назначает логический номер LUN физическому устройству. Это означает, что задача по запросам через данный логический номер будет использовать назначенное физическое устройство. Вызов на языке Фортран:

```
CALL ASNLUN (лгн, иустр, нустр [,кдз])
```

Макровызов:

```
ALUN⊙ лгн, иустр, нустр
```

Директива CINT⊙ — подключиться к вектору прерывания — позволяет задаче самой обрабатывать прерывания от устройства. Программа обслуживания прерываний (ISR-программа) размещается в теле задачи. В системе с диспетчером памяти эту директиву может издавать только привилегированная задача.

Прерывание может обслуживаться на трех различных уровнях: на уровне прерывания устройства; на уровне отложенного прерывания и на уровне задачи.

На уровне задачи различаются два подуровня: AST-прерывание и не AST-прерывание. Вызов на языке Фортран отсутствует.

Макровызов:

```
CINT⊙ век, баз, адрпр, рзадр, прт, аст
```

Для отключения от вектора прерываний аргумент адрпр должен быть равен нулю; аргументы баз, рзадр и аст в этом случае игнорируются.

Директива GLUN⊙ — предоставить информацию об устройстве с заданным логическим номером — заполняет 6-словный буфер информацией о физическом устройстве, логический номер (LUN) которого указан в директиве. Возвращаемая информация будет включать имя, номер и характеристики физического устройства. Вызов на языке Фортран:

```
CALL GETLUN (лгн, буф[,кдз])
```

Макровызов:

```
GLUN⊙ лгн, буф
```

Директива GMCR⊙ — дать командную строку — позволяет передать 80-байтную командную строку задаче, выполнившей директиву. Если задача установлена с именем «...имя» или «имя T## #», где имя состоит из трех алфавитно-цифровых символов, ## — восьмеричный номер терминала, с которого введена команда:

> имя командная строка

то программа связи с оператором будет запрашивать выполнение этой задачи. Задача, вызванная таким способом, должна издать директиву GMCR⊙, которая передаст командную строку в 80-байтный буфер задачи. Вызов на языке Фортран:

CALL GETMCR (буф [,кдз])

Макровызов:

GMCR

⊙S — форма директивы GMCR не обеспечивается, так как командная строка принимается в блок DPB. Система преобразует командную строку следующим образом: заменяет код табуляции на пробел; заменяет несколько пробелов одним; удаляет все концевые пробелы. Последним символом в строке является ограничитель строки ⟨CR⟩ или ⟨ESC⟩.

Директива QIO⊙ — поставить в очередь запрос ввода-вывода — ставит требование ввода-вывода задачи в очередь к указанному физическому устройству, упорядоченную по приоритетам задач. Физическое устройство задается логическим номером устройства (LUN).

По завершении операции ввода-вывода управляющая программа объявляет важное событие. Если в директиве указан флаг события, то при включении требования в очередь он обнуляется. При завершении обработки этого запроса флаг устанавливается, кроме того, когда требование ставится в очередь, очищается блок состояния ввода-вывода. После выполнения этого запроса в него засылается код завершения операции ввода-вывода.

Если адрес AST-программы задан, то по завершении ввода-вывода произойдет прерывание задачи; AST-программа получит в стеке слово маски ожидаемых флагов событий, PS и PS задачи в точке AST-прерывания, слово состояния директивы и адрес блока состояния ввода-вывода. Вызов на языке Фортран:

CALL QIO (кдф , лгн , [флг] , [прт] , [ствв] , [прм] [,кдз])

где прм — 6-словный целый массив параметров, зависящих от устройства, которые в DPB занимают 1—6 слов параметров;

прт — приоритет; игнорируется, но должен быть указан запятой.

Макровызов:

QIO⊙ кдф , лгн , [флг] , [прт] , [ствв] , [аст] , [прм]

где прм — список параметров в виде ⟨P1,..., P6⟩, зависящих от устройства;

прт — приоритет; он игнорируется, но место параметра должно быть указано запятой.

Директива QIOW⊙ — поставить в очередь запрос ввода-вывода и ждать его завершения — подобна директиве QIO⊙ за исключением того, что если в директиве QIOW⊙ указан флаг события, то к директиве QIO⊙ автоматически добавляется директива ждать единичный флаг (WTSE⊙). Если флаг не задан, то директива QIOW⊙ выполняется так же, как и директива QIO⊙. Вызов на языке Фортран:

CALL WTQIO (кдф , лгн , [флг] , [прт] , [ствв] , [прм] [,кдз])

Макровызов:

QIOW⊙ кдф , лгн , [флг] , [прт] , [ствв] , [аст] , [, прм]

Директива RCVD⊙ — получить данные — получает из очереди данных, посланных задаче, 13-словный блок, который был поставлен в очередь другой задачей с помощью директивы SDAT⊙ — послать данные. Дисциплина обработки очереди: «первым пришел, первым обслужен». RCVD⊙ передает задаче в 15-словном буфере имя посылающей задачи (два слова в RAD-50) и 13—слов данных, переданных директивой SDAT⊙. Подчиненная задача, если только у нее нет активных групповых глобальных флагов, получает по директиве RCVD⊙ код UIC и терминал, назначенный к устройству TI: задачи, издавшей директиву SDAT. Вызов на языке Фортран:

CALL RECEIV ([имя], буф [,,кдз])

где буф — 15-словный целый массив для получения данных.

Макровызов:

RCVD⊙ [имя] , буф

где буф — адрес 15-словного буфера для получения данных.

Если имя посылающей задачи указано, то директива передает данные, посланные соответствующей задачей; если имя не задано, то от любой задачи.

Директива RCVX⊙ — *получить данные или завершить задачу* — получает из очереди данных, посланных задаче, 13-словный блок, который был поставлен в эту очередь

другой задачей с помощью директивы SDAT⊙ (послать данные), либо завершает задачу, если данные не были посланы. Дисциплина обработки очереди: «первым пришел, первым обслужен». Имя посылающей задачи (два слова в RAD-50) и 13-словный блок данных передаются в 15-словный буфер, указанный в директиве; имя задачи занимает первые два слова. Чтобы не потерять данные, посланные задаче во время ее завершения, пользователь должен позаботиться о быстром завершении задачи. Рекомендуется дождаться завершения всех операций ввода-вывода и закрытия файлов до выдачи директивы RCVX⊙.

Подчиненная задача, если только у нее нет активных групповых глобальных флагов, получает по директиве RCVX⊙ код UIC и терминал TI: задачи, издавшей директиву SDAT. Вызов на языке Фортран:

```
CALL RECOEX ([имя] , буф [, ,кдз])
```

Макровызов:

```
RCVX⊙ [имя], буф
```

Директива SDAT⊙ — *послать данные* — ставит в очередь данных получающей задачи 13-словный блок данных (дисциплина обработки очереди «первым пришел, первым обслужен») и объявляет важное событие. Если указан локальный флаг события, то он устанавливается для посылающей задачи. Вызов на языке Фортран:

```
CALL SEND (имя , буф , [флг] [,кдз])
```

где буф — 13-словный целый массив для посылки данных

Макровызов:

```
SDAT⊙ имя , буф [,флг]
```

где буф — адрес 13-словного буфера данных.

2.3.7. ДИРЕКТИВЫ УПРАВЛЕНИЯ ПАМЯТЬЮ

Директивы управления памятью позволяют задаче манипулировать виртуальным и логическим адресным пространством, динамически создавать и контролировать состояние районов и адресных окон. Директивы этой группы позволяют задачам обмениваться данными, передавая друг другу информацию о размещении данных в районе.

Директива ATRG⊙ — присоединить район — присоединяет к задаче статический или динамический район. Управляющая программа проверяет совместимость за-

прошенного типа доступа, заданного в слове состояния района, с правами на доступ к району, определенному для той категории пользователей, к которой относится задача. Если запрошенный тип доступа разрешен, то район присоединяется к задаче, и управляющая программа возвращает в поле R.GID блока описания района (RDB) 16-битный идентификатор района (ID района). ID района используется задачей в последующих директивах управления памятью. Если район уже присоединен к задаче, то директива ATRG⊙ может использоваться только для получения ID района. Вызов на языке Фортран:

```
CALL ATRG (бор [,кдз])
```

Макровызов:

```
ATRG⊙ бор
```

Директива CRAW⊙ — *создать адресное окно* — создает новое виртуальное адресное окно: заполняет блок окна в заголовке задачи, задавая виртуальный базовый адрес и размер окна. Место для блока окна в заголовке задачи резервируется при построении задачи по параметру WNDWS. Если новое окно перекрывает ранее созданные окна, то для старых окон прекращается отображение, и они уничтожаются. При успешном создании окна задаче возвращается 8-битный идентификатор окна (ID окна). ID окна, возвращаемый задаче, — это число от 1 до 15; ID окна является индексом блока окна в заголовке задачи. Блок окна описывает созданное адресное окно. Вызов на языке Фортран:

```
CALL CRAW (боо [,кдз])
```

Макровызов:

```
CRAW⊙ боо
```

Директива CRRG⊙ — создать район — создает динамический район в системно-управляемом разделе и в зависимости от параметров может присоединить его к задаче, выполнившей директиву. Управляющая программа возвращает код ошибки, если для размещения района в указанном разделе недостаточно места. Вызов на языке Фортран:

```
CALL CRRG (бор [,кдз] )
```

Макровызов:

```
CRRG⊙ бор
```

Директива DTRG⊙ — отсоединить район — отсоединяет ранее присоединенный район от задачи, выполнившей директиву. При этом автоматически прекращается отображение всех окон, отображаемых в текущий момент в район. Вызов на языке Фортран:

CALL DTRG (бор [,кдз])

Макровызов:

DTRG⊙ бор

Директива ELAW⊙ — уничтожить адресное окно — уничтожает указанное адресное окно. Если в момент уничтожения производится отображение адресного окна, то отображение прекращается. Последующее использование идентификатора уничтоженного окна недопустимо. Вызов на языке Фортран:

CALL ELAW (боо [,кдз])

Макровызов:

ELAW⊙ боо

Директива GMCX⊙ — получить описание окон — возвращает описание существующих в текущий момент адресных окон. Описание возвращается в формате, позволяющем пользователю восстановить отображение адресных окон с помощью директив CRAW⊙ в формате блоков описания окон WDB. Аргументом GMCX⊙ является адрес таблицы, содержащей свободные блоки окон WDB, и слово-ограничитель таблицы (для каждого блока окна, размещенного в заголовке задачи HDR, должен быть задан свой блок WDB). Вызов на языке Фортран:

CALL GMCX (тбл [,кдз])

где тбл — целый массив для получения описания существующих в текущий момент адресных окон; размер массива равен $8 \times N + 1$, где N — число блоков окон в HDR; максимальный размер массива равен $8 \times 8 + 1 = 65$ словам;

Макровызов:

GMCX⊙ тбл

где тбл — адрес таблицы, состоящей из N блоков WDB и слова-ограничителя; N — число блоков окон в HDR.

Директива MAP⊙ — отобразить адресное окно — отображает адресное окно в присоединенный район. Отображение начинается с заданного места района. Если

окно уже отображается, то, прежде чем выполнить запрошенное отображение, управляющая программа прекращает действующее отображение.

Длина, задаваемая для отображения, должна быть меньше или равна минимальной из следующих величин: размера окна, указанного при создании окна; длины, оставшейся между указанным началом отображения в районе и концом района.

Чтобы иметь возможность отображать окно с правом доступа на запись, район должен быть присоединен с правом доступа на запись. Чтобы можно было выполнять отображение с правом доступа на чтение, район должен быть присоединен с правом доступа на чтение либо на запись. Вызов на языке Фортран:

```
CALL MAP (боо [,кдз] )
```

Макровызов:

```
MAP⊙ боо
```

Директива RREF⊙ — получить по ссылке — выбирает и передает задаче элемент из очереди получений по ссылке получающей задачи. В зависимости от заданных параметров задача может завершиться, если очередь получений по ссылке окажется пустой. В директиве может быть указано, что управляющая программа должна выполнить отображение виртуального адресного окна в область, в которой передаются данные. Если директива завершена успешно, то объявляется важное событие. Каждый элемент очереди, получений по ссылке, требует отдельного присоединения района к получающей задаче. Район, в который передано несколько посылок для задачи, присоединяется к ней соответствующее число раз. Так как присоединение района требует системной динамической памяти, то для ее экономии получающая задача должна предварительно отсоединять ранее присоединенный ею район; то есть необходимо, чтобы каждый район был присоединен к задаче только один раз. Вызов на языке Фортран:

```
CALL RREF (боо, [буф] [,кдз] )
```

где буф — 10-словный целый массив, используемый как дополнительный буфер данных.

Макровызов:

```
RREF⊙ боо
```

Директива SREF⊙ — послать данные посредством передачи ссылки —ставит элемент, содержащий информа-

цию, необходимую для обращения к району, в очередь получений по ссылке указанной (получающей) задачи. По каждой директиве SREF⊙, изданной задачей, управляющая программа автоматически присоединяет район, на который делается ссылка, к получающей задаче. Присоединение происходит, даже если этот район уже присоединен к получающей задаче. Вызов на языке Фортран:

CALL SREF (имя, [флг], боо, [инф] [,кдз])

где инф — 8-словный целый массив, содержащий дополнительную информацию. Если такой массив задан, то его адрес содержится в восьмом слове блока описания окна. Если этот адрес опущен, то содержимое восьмого слова блока описания окна не изменяется.

Макровызов:

SREF⊙ имя, боо [,флг]

где имя — имя получающей задачи.

Директива UMAP⊙ — прекратить отображение адресного окна — прекращает отображение заданного адресного окна. После того как отображение окна прекращено, обращение к виртуальным адресам окна недопустимо и вызывает прерывание процессора. Вызов на языке Фортран:

CALL UMAP (боо [,кдз])

Макровызов:

UMAP⊙ боо

2.3.8. ДИРЕКТИВЫ ПОРОЖДЕНИЯ ЗАДАЧ

Для синхронизации выполнения задач реального времени большую роль играет механизм, устанавливающий связь задач друг с другом, основанный на понятиях «задача-родитель» и «задача-потомок». *Задача-родитель* — это задача, которая либо «порождает» другую задачу, либо «подключается» к другой задаче, называемой *задачей-потомком.*

Порождение задачи — это активизация одной задачи из другой и установление связи между этими задачами, называемыми «родителем» и «потомком». *Подключение к задаче* — это только установление связи между двумя активными задачами — родителем и потомком. Связь «родитель-потомок» подразумевает, что задача-родитель может быть оповещена о завершении задачи-потомка.

Кроме того, задача-родитель получает код завершения задачи-потомка либо может получить информацию о текущем состоянии потомка. Возвращаемый код состояния сообщает об успешном завершении потомка либо указывает на возникновение ошибочной ситуации при выполнении задачи-потомка. Отношение «родитель-потомок» полезно для организации пакетной обработки.

Директивы межзадачных отношений «родитель-потомок». Директивы, порождающие между задачами отношения «родитель-потомок», делятся на две группы: директивы, создающие эти отношения, и директивы, передающие отношения «родитель-потомок» другой задаче.

Для создания отношений «родитель-потомок» используются три директивы — SPWN⊙, CNCT⊙, SDRC⊙. *Директива SPWN⊙ — породить задачу* — директива активизирует задачу-потомка и подключается к ней. В отличие от запуска задач по директивам RUN⊙ или RQST⊙ активизация задачи по директиве SPWN⊙ имеет следующие преимущества:

можно передавать порожденной задаче командную строку;

порожденная задача может вернуть задаче-родителю (родителям) информацию о своем текущем состоянии либо о состоянии при завершении.

Директива CNCT⊙ — подключиться к задаче — устанавливает связь между активными задачами для синхронизации выполнения задачи-родителя. Синхронизация выполняется за счет возврата задаче-родителю текущего состояния либо кода завершения задачи-потомка.

Директива SDRC⊙ — послать данные, запросить выполнение, подключиться — посылает данные указанной задаче, издает запрос на ее выполнение, если задача не активна, и подключает задачу, издавшую директиву, к указанной задаче.

Группа директив, передающих отношения между задачами. Эта группа включает следующие директивы: RPOI⊙, SDRP⊙.

Директива RPOI⊙ — запросить выполнение и передать информацию о подключении — позволяет задаче-потомку передать своих «родителей» другой задаче. Таким образом, родители задачи, издавшей директиву RPOI⊙, получают нового потомка.

Директива SDRP⊙ — послать данные, запросить выполнение, передать ОСВ — посылает задаче данные, передает ей своих «родителей» и запрашивает задачу на выполнение, если она не активна.

С помощью директив SPWN⊙ и CNCT⊙ задача может подключиться к нескольким задачам-потомкам. Используя эти директивы, задача-родитель может многократно подключиться к задаче-потомку. В свою очередь, к задаче-потомку могут подключиться несколько задач-родителей. При каждом подключении одной задачи к другой создается специальный управляющий блок — блок управления порождением (ОСВ).

Директивы связи задач. Задача-потомок может вернуть задаче-родителю информацию о своем состоянии с помощью двух директив:

EXST⊙ — завершить задачу и выдать состояние и EMST⊙ — выдать состояние.

Директива EXST⊙, изданная задачей-потомком, вызывает завершение этой задачи и передачу ее состояния задаче (или всем задачам), подключенным к потомку с помощью директив SPWN⊙, CNCT⊙ или SDRC⊙.

Директива EMST⊙ издается задачей-потомком и вызывает передачу состояния задаче-родителю, указанному в директиве, либо, если родитель явно не указан, — всем задачам-родителям.

После возврата состояния задачи-потомка задачам-родителям подключенные к задаче родители отключаются.

Директива CNCT⊙ — подключиться к задаче — позволяет синхронизировать выполнение задачи, издавшей директиву, с завершением или с изменением состояния другой задачи, активной в момент издания директивы CNCT⊙, которая устанавливает связь с другой задачей (задачей-потомком), создает блок управления порождением (ОСВ) и ставит его в очередь к задаче-потомку. Кроме того, CNCT⊙ увеличивает на единицу счетчик задач-потомков в ТСВ задачи (T. RDCT), издавшей директиву. Счетчик потомков указывает число задач, к которым подключена данная задача. При завершении задачи-потомка или при выдаче ею состояния для задачи-родителя издается AST-прерывание; соответствующая AST-программа получает на входе в стеке задачи адрес блока состояния, содержащего код состояния задачи-потомка. Эта директива не должна издаваться для подключения к задаче, являющейся интерпретатором командных строк. Вызов на языке Фортран:

CALL CNCT (имя, [флг], [аст], [блс],]прм] [,кдз])

где имя — переменная с плавающей точкой единичной точности, содержащая имя задачи в RAD-50, к которой подключается задача, издавшая директиву;

блс—имя 8-словного блока состояния, который заполняется при завершении задачи-потомка или при выдаче ею состояния. Содержит код завершения задачи-потомка и код аварийного завершения для задачи вывода системных сообщений (TKTN);

прм — имя слова, в котором передается адрес блока состояния задачи-потомка для AST-программы.

Макровызов:

CNCT⊙ имя , [флг] , [аст] , [блс]

где имя—имя задачи, к которой подключается задача, издавшая директиву.

Директива EXST⊙ — выдать состояние — передает указанной задаче-родителю информацию о состоянии задачи: возвращает заданную в параметрах 16-битную величину задаче-родителю. Директива устанавливает также флаг и (или) объявляет AST-прерывание, если при подключении к данной задаче по директивам SDRC⊙ (послать данные, запросить выполнение, подключиться), SPWN⊙ (породить задачу), CNCT⊙ (подключиться к задаче), флаг и (или) AST-программа были заданы.

Если указанная в EMST⊙ задача многократно подключалась к задаче, издавшей директиву, то для возвращения состояния задаче-родителю используется первый (самый старый) из соответствующих блоков управления порождением (ОСВ). Если в директиве имя задачи опущено, то указанные выше действия выполняются для всех задач, подключенных в данный момент к задаче, издавшей директиву. После передачи информации о состоянии задачи задаче-родителю (родителям) директива уничтожает подключение. Вызов на языке Фортран:

CALL EMST ([имя] , сост[,кдз])

где имя — имя задачи-родителя, которой задача передает информацию о своем состоянии;

сост — 16-битная величина, передаваемая задаче-родителю.

Макровызов:

EMST⊙ [имя] , сост

Директива EXST⊙ — завершить задачу и выдать состояние завершает выполнение задачи, издавшей эту директиву, и передает 16-битный код завершения задачи (состояние задачи) всем задачам, подключенным к ней по директивам SPWN⊙, CNCT⊙ или SDRC⊙. Если подключенных задач нет, то EXST⊙ завершает

выполнение задачи. Управляющая программа не накладывает никаких ограничений на формат кода состояния: этот формат известен только задаче-родителю и задаче-потомку. Однако, если задача-потомок по какой-то причине аварийно завершается, то подключенным к ней задачам возвращается код EX⊙SEV — код грубой ошибки. В тех случаях, когда задача издает обычную директиву завершения, всем задачам, подключенным к ней, передается код EX⊙SUC — код успешного завершения. Вызов на языке Фортран:

```
CALL EXST (сост)
```

Макровызов:

```
EXST⊙ сост
```

Директива RPOI⊙ — запросить выполнение и передать информацию о подключении — запрашивает выполнение указанной задачи и подключает к ней одного или всех родителей задачи, издавшей директиву.

Директива позволяет передать командную строку указанной задаче. По директиве RPOI⊙ привилегированные задачи и интерпретаторы командных строк (CLI) могут передать запрашиваемой задаче также код идентификации пользователя UIC и терминал, назначаемый в качестве устройства TI: запрашиваемой задачи. Вызов на языке Фортран:

```
CALL RPOI⊙ (имя , [гр] , [чп] , [имяр] , [буф] , [длбуф] ,
           [флб] , [иустр] , [нустр] , [имяз] , [блп] [,кдз])
```

где имя — имя массива, содержащего имя задачи (в RAD-50), которая запрашивается на выполнение;

имяр — имя массива или целая переменная формата I*4, содержащего имя задачи-родителя;

флб — флажковый байт, задающий функции директивы: завершить задачу в случае успешного выполнения директивы;

передать всех родителей задачи, издавшей директиву, запрашиваемой задаче. По умолчанию не передавать. Задаче CLI передавать родителей нельзя;

передать запрашиваемой задаче первого родителя из очереди задач, подключенных к издавшей директиву задаче (если он имеется);

нустр — имя целой переменной, содержащей имя устройства (в символьном коде), передаваемого запрашиваемой задаче в качестве TI:;

имяз — имя массива, содержащего имя (в RAD-50), под которым будет выполняться запрашиваемая задача. Этот параметр может быть указан только задачей CLI;

блп — имя целой переменной, содержащей адрес блока управления порождением ОСВ, передаваемого запрашиваемой задаче. Этот адрес может быть получен только через фортрановскую подпро-

грамму GTCMCI в информационном буфере. Данный параметр допустим только для задач CLI.

Макровызов:

RPOI⊙ имя , , , [гр] , [чл] , [имяр] , [буф] , [длбуф] , [флб] , [иустр] , [нустр] , [имяз] [,блп]

Директива SDRC⊙ — послать данные, запросить выполнение, подключиться — обеспечивает следующие функции: посылку данных указанной задаче, запрос этой задачи на выполнение, если она еще не активна, и подключение к ней. Получающая задача обычно возвращает код своего состояния через директиву EMST⊙ (выдать состояние) или EXST⊙ (завершить задачу и выдать состояние). Вызов на языке Фортран:

CALL SDRC (имя , буф , [флг] , [аст] , [блс] , [прм] [,кдз])

где имя — имя запрашиваемой задачи, к которой подключается задача, издавшая директиву;

буф — имя 13-словного буфера;

блс — имя 8-словного блока состояния, заполняемого при завершении задачи-потомка или при возврате его состояния. Может содержать: код состояния задачи потомка и код аварийного завершения для задачи вывода системных сообщений (TKTN);

прм — имя слова, в котором передается адрес блока состояния для AST-программы;

Макровызов:

SDRC⊙ имя , буф , [флг] , [аст] , [блс]

Директива SDRP⊙ — послать данные, запросить выполнение, передать OCB — посылает указанной задаче данные, запрашивает эту задачу на выполнение, если она еще не активна, и передает ей своих родителей. Вызов на языке Фортран:

CALL SDRP (имя , буф , [длбуф] , [флг] , [флб] , [имяр] , [блп] [,кдз])

Макровызов:

SDRP⊙ имя , буф , [длбуф] , [флг] , [флб] , [имяр] , [блп]

Директива SPWN⊙ — породить задачу — выполняет следующие функции: издает запрос на выполнение указанной задачи, ставит в очередь командных строк переданную в директиве строку, назначает терминал для устройства TI: активизируемой задачи. Директива SPWN⊙ выполняет порождение указанной задачи: создает блок управления порождением OCB, ставит его в очередь OCB к указанной (порожденной) задаче, увели-

чивает счетчик порожденных задач в TCB задачи, издавшей директиву.

Счетчик порожденных задач указывает управляющей программе, что задача является задачей-родителем и имеет одного или более потомков. Если задача-родитель в момент завершения подключена к одной или более задачам, то счетчик порожденных задач очищается. При завершении задачи-потомка счетчик порожденных задач уменьшается на единицу.

Блок OCB содержит адрес TCB задачи-родителя, а также всю необходимую информацию для обработки события, связанного с завершением порожденной задачи.

Если в директиве задана командная строка, то строка сохраняется в системной динамической области и ставится в системную очередь командных строк. Порожденная задача получает командную строку по директиве GMCR⊙ (получить командную строку). Максимальная длина командной строки равна 79 байтам.

Если в директиве задан адрес AST-программы, то по завершении порожденной задачи для задачи-родителя объявляется асинхронное прерывание. На входе AST-программа получает в стеке задачи адрес блока состояния, содержащего код завершения порожденной задачи. Перед входом в директиву ASTX⊙ (выход из программы обслуживания AST) AST-программа должна удалить из стека адрес блока состояния задачи.

Если порожденная задача является интерпретатором командных строк, то директива SPWN⊙ выполняет специальные действия. Командная строка (она должна быть обязательно указана в директиве) и блок OCB ставятся в очередь к задаче-интерпретатору, который либо сам обрабатывает командную строку, либо передает другой задаче. В том случае, когда командная строка обрабатывается самим интерпретатором, блок OCB используется, чтобы имитировать немедленное завершение интерпретатора и передать породившей задаче через соответствующую управляющую подпрограмму код завершения интерпретатора. Если же интерпретатор передает командную строку другой задаче, то он просто переставляет блок из своей очереди в очередь OCB к той задаче, которой предназначена командная строка. Командная строка переставляется в очередь командных строк задачи, как если бы она была послана задаче непосредственно. Таким образом, задача, фактически получающая командную строку, находится в таком же состоянии, как если бы она была явно указанная

в директиве SPWN⊙. Условия завершения порожденной задачи не возникают до тех пор, пока задача, фактически получившая командную строку, не завершится. Вызов на языке Фортран:

```
CALL SPAWN (имя , [гр] , [чл] , [флг] , [аст] , [блс] , [прм] ,
            [стр] , [длстр] , [нустр] , [иустр] [,кдз])
```

Макровызов:

```
SPWN⊙ имя , , , [пр] , [чл] , [флг] , [аст] , [блс] , [стр] , [длстр] ,
      [нустр] , [иустр]
```

2.3.9 ДИРЕКТИВЫ, ПРЕДНАЗНАЧЕННЫЕ ДЛЯ CLI

Эта группа директив позволяет задачам, являющимся интерпретаторами командных строк, получать командные строки, запрашивать и передавать информацию о подключении, получать информацию от интерпретатора, назначать CLI терминалу.

Директива GCCI⊙ — дать команду для CLI — передает командную строку интерпретатору командных строк (CLI) и переписывает ее из системного буфера в буфер задачи. Задаче также передается информация о терминале, с которого подана команда. Директиву GCCI⊙ могут использовать только задачи, являющиеся интерпретаторами командных строк. Вызов на языке Фортран:

```
CALL GTCMCI(буф , длбуф , [ибуф] , [длибуф] , [адрком] ,
     [мск] , [,кдз] )
```

где буф — имя байтного массива (командного буфера), куда передается командная строка;

длбуф — имя целой переменной, содержащей длину командного буфера в байтах;

ибуф — имя целого массива (информационного буфера), куда передается вспомогательная информация для CLI;

длибуф — имя целой переменной, содержащей длину информационного буфера. Если длина меньше размера передаваемой информации, то в буфер передается столько информации, какова длина, указанная в длибуф;

адрком — имя целой переменной, содержащей адрес системного буфера, в котором лежит команда для CLI;

мск — имя целой переменной, содержащей битовую маску, которая указывает, какие действия предпринять управляющей программе, если очередь командных строк пуста. Могут быть запрошены функции:

вернуть управление задаче с установкой бита C в PSW (умолчание);

завершить CLI вместо возврата управления в CLI;

перевести CLI в останов вместо возврата управления в CLI;

передать команду в буфер задачи, но не удалять ее из системной очереди.

В программах на Фортране эти биты должны быть заданы десятичными величинами.

Макровызов:

GCCI⊙ буф, длбуф, [ибуф], [длибуф], [адрком], [мск]

где буф — адрес буфера, куда передается командная строка;

длбуф — длина командного буфера. Максимальная длина буфера равна 91 байту;

ибуф — адрес буфера для получения информации о терминале, с которого подана команда;

длибуф — длина информационного буфера;

адрком — адрес команды в системном буфере;

мск — битовая маска, задающая действия управляющей программы, если очередь командных строк пуста. Могут быть запрошены функции:

вернуть управление задаче с установкой бита С в PSW (умолчание);

завершить CLI вместо возврата управления в CLI;

перевести CLI в останов вместо возврата управления в CLI;

передать команду в буфер задачи, но не удалять ее из системной очереди;

Директива GCLI⊙ — дать информацию о CLI — передает в буфер задачи информацию об указанном интерпретаторе командных строк (CLI) либо о CLI, связанном с указанным терминалом. Только привилегированная задача может указать интерпретатор, не связанный с устройством TI: задачи, или терминал, не являющийся устройством TI: задачи. Вызов на языке Фортран:

CALL GETCLI (буф, длбуф, [имяи], [иустр], [нустр] [,кдз])

где буф — имя целого массива (буфера), куда посылается информация о CLI;

длбуф — длина буфера в байтах;

имяи — имя двухсловного массива, содержащего имя интерпретатора командных строк в RAD-50;

иустр — имя целой переменной, содержащей имя терминала в символьном коде (по умолчанию — устройства TI:);

нустр — имя целой переменной, содержащей восьмеричный номер терминала;

Макровызов:

GCLI⊙ буф, длбуф, [имяи], [иустр], [нустр]

где буф — адрес буфера, куда посылается информация о CLI;

длбуф — длина буфера;

имяи — имя CLI в RAD-50;

иустр — имя терминала (в символьном коде), о CLI которого запрашивается информация (по умолчанию устройства TI:);

нустр — восьмеричный номер терминала.

Директива SCLI⊙ — назначить CLI терминалу — назначает указанный интерпретатор командных строк (CLI)

в качестве CLI указанного терминала. Директива SCLI⊙ может быть издана только привилегированной задачей или задачей CLI. Если в слове состояния CLI (в блоке описания CLI) указан флаг ограничения (CP. RST), то только сама задача может установить себя терминалу. Вызов на языке Фортран:

CALL SETCLI (имя , иустр , нустр [,кдз])

где имя — имя двухсловного массива, содержащего имя интерпретатора командных строк, назначаемого терминалу;

иустр — имя целой переменной, содержащей имя терминала, к которому устанавливается CLI (в символьном коде); по умолчанию — устройство TI:;

нустр — имя переменной, содержащей номер терминала;

Макровызов:

SCLI⊙ имя , [иустр] , [нустр]

где имя — имя CLI, устанавливаемого к терминалу;

иустр — имя терминала (в символьном коде), к которому устанавливается интерпретатор; по умолчанию — устройство TI:

нустр — номер терминала.

3

СИСТЕМА УПРАВЛЕНИЯ ФАЙЛАМИ ОСРВ

Система управления файлами FCS позволяет выполнять операции ввода-вывода, ориентированные на записи или на блоки, и осуществляет дополнительные функции, необходимые для управления файлами (например, открытие, закрытие, ожидание, удаление). Система управления файлами осуществляет доступ к внешним наборам данных, не зависимый от устройств.

Система управления файлами ОСРВ включает в себя *вспомогательные управляющие процессоры* ACP, поддерживающие файловые структуры на внешних носителях, *набор системных объектных модулей* (подпрограмм), реализующих алгоритмы обработки файлов, а также *группу специальных макровызовов*, обеспечивающих интерфейс между пользовательской программой и FCS и генерирующих необходимые при выполнении всех операций ввода-вывода структуры данных.

Структуры данных включают блок описания файла FDB, содержащий необходимую для обработки файла информацию, блок описания набора данных, используемый FCS при открытии файла для получения спецификации файла в символьном коде, блок имени файла по умолчанию, дополняющий отсутствующие поля спецификации файла, и файловую область (FSR), используемую FCS как рабочую память.

FCS поддерживает последовательный и прямой методы доступа к записям. *Последовательный метод доступа* не зависит от устройства, *прямой метод доступа* может быть использован только для магнитных дисков.

Данные передаются блоками между внешними устройствами и памятью. Блок может содержать одну или несколько *логических записей.* Допускаются *записи*

фиксированной или *переменной длины,* которые, в свою очередь, могут быть *переходящими* (пересекать границы блоков) или *непереходящими.*

На устройствах прямого доступа записи переменной длины могут быть двух типов. *Последовательным записям переменной длины* предшествуют двухсловные заголовки (первое слово содержит длину записи, второе слово — номер записи). *Произвольным записям переменной длины* предшествует однословный заголовок, содержащий длину записи.

Блоки и записи в файлах последовательно пронумерованы, начиная с 1. *Виртуальный номер блока* есть число, определяющее положение блока относительно начала файла, в отличие от *логического номера блока,* определяющего положение блока относительно начала тома.

Система управления файлами допускает два способа обмена: виртуальными блоками и записями. В первом случае *единицей обмена* служит виртуальный блок, который пересылается между внешним устройством и памятью пользовательской программы. Блокирование (упаковка и распаковка) записей в этом случае не происходит. Обмен блоками выполняется асинхронно с работой пользовательской программы. Во втором случае единицей обмена служит логическая запись. Виртуальные блоки пересылаются между внешним устройством и файловой областью (блочными буферами), в которой производится упаковка и распаковка записей. Обмен записями всегда выполняется синхронно, т. е. пользовательская программа получает управление после завершения обмена.

При использовании способа обмена записями допускаются два режима передачи данных. В *режиме пересылки* запись пересылается между блочным буфером и буфером записи в пользовательской программе, где и обрабатывается программой. В *режиме указания места* запись обрабатывается непосредственно в блочном буфере, а пересылка ее нужна только тогда, когда запись пересекает границу блока.

Система управления файлами FCS обеспечивает возможность читать данные в буфере, предваряя требования программы пользователя (предварительный ввод), и писать содержимое буферов, в то время как пользовательская программа готовит новые записи для вывода (довывод). Такое совмещение обработки данных с операциями ввода-вывода, называемое *многоуровневой буфери-*

зацией, достигается за счет выделения в файловой области памяти не одного, а нескольких блочных буферов.

Система управления файлами допускает одновременный доступ к файлам из одной или нескольких задач. При этом пользователь сам определяет допустимость конкурирующих запросов других задач на запись в файл, открытый им для чтения.

Более подробная информация о файловой структуре ОСРВ на магнитных дисках содержится в приложениях 2 и 3, а файловая структура ОСРВ на магнитной ленте описана в приложении 4.

В приложении 5 приведены структура и полное описание полей блока описания файла FDB, а в приложении 6 — блока имени файла.

3.1. ДИРЕКТИВЫ УПРАВЛЕНИЯ ВВОДОМ-ВЫВОДОМ

3.1.1. ОБЩИЕ СОГЛАШЕНИЯ ПО ОПИСАНИЮ ФОРМАТОВ МАКРОВЫЗОВОВ

В целях унификации описания форматов макровызовов директив управления вводом-выводом введем перечень соглашений, используемых при описании их синтаксиса.

1. Заглавные буквы означают, что данное поле формата должно быть заполнено именно так, как показано. Строчные буквы указывают обозначения параметров, а не их значения.

2. Все параметры всех директив являются необязательными. При опускании параметров в макровызовах для создания управляющих блоков системы FCS соответствующие поля получают значения по умолчанию. Большинство параметров макровызовов обработки файлов вызывают заполнение информацией полей блока FDB, отсутствие этих параметров в макровызове не изменяет значение соответствующих полей.

3. При описании форматов используются следующие единые обозначения параметров:

actl — указание специфических параметров обработки файла.

В младший байт поля F.ACTL заносится число указателей извлечения, которые должны храниться в динамической памяти для открытого файла. По умолчанию используется значение, указанное в ключе WIN команд оператора INITVOLUME или MOUNT; Значением параметра может быть:

FA.ENB — отказ от значений специфических параметров по умолчанию;

FA.POS — отказ от позиционирования на конец тома магнитной ленты при открытии выводного файла;

FA.RWD — перемотка тома магнитной ленты перед открытием и после закрытия файла;

FA.DLK — открытый для записи файл на диске не должен блокироваться, если он не закрыт;

aloc — число со знаком, определяющее количество блоков диска, на которое файл будет увеличиваться при расширении во время записи. Положительное значение указывает, что файл должен остаться непрерывным, отрицательное — файл может быть прерывистым;

bkda — адрес буфера для операций ввода-вывода блоков;

bkdn — адрес подпрограммы обработки асинхронного системного прерывания;

bkds — размер (в байтах) блока, который должен быть считан или записан;

bkef — номер флага событий для синхронизации ввода-вывода блоков;

bkst — адрес двухсловного блока состояния ввода-вывода блоков;

bkvb — адрес двухсловного блока, содержащего виртуальный номер блока в операциях READ⊙ и WRITE⊙;

cntg — число со знаком, определяющее количество блоков диска, которое нужно выделить файлу при его создании. Положительное значение указывает, что создается непрерывный файл, отрицательное — создается прерывистый файл;

dfnb — адрес блока имени файла по умолчанию. Этот блок может быть создан по макровызову NMBLK⊙;

dspt — адрес блока описания набора данных. Блок описания набора данных — это пользовательская 6-словная структура данных, включающая в себя описатели строк имени устройства, кода идентификации каталога и имени файла. Описатель строки состоит из двух слов — первое слово длина, второе слово адрес соответствующей строки;

efn — номер флага событий для синхронизации ввода-вывода записей;

err — адрес подпрограммы обработки ошибок. Если во время операции обработки файла обнаружена ошибка, то устанавливается бит C слова состояния процессора, в поле F.ERR блока FDB заносится код ошибки и выполняется инструкция CALL для передачи управления на указанную подпрограмму;

facc — значением параметра может быть:

FO.RD — существующий файл должен быть открыт для чтения;

FO.WRT — должен быть создан новый файл и открыт для записи;

FO.APD — существующий файл должен быть открыт для дополнения;

FO.MFY — существующий файл должен быть открыт для перезаписи без изменения его длины;

FO.UPD — существующий файл должен быть открыт для записи и расширения;

FA.NSP — в комбинации с FO.WRT старый файл с такой же полной спецификацией не должен быть заменен новым файлом;

FA.TMP — в комбинации с FO.WRT должен быть создан временный файл;

FA.SHR — файл должен быть открыт для разделенного доступа;

fdb — адрес блока описания файла, к которому относится соответствующий макровызов. Если параметр опущен, то предполагается, что адрес FDB находится в регистре R0;

lun — логический номер устройства, на котором расположен файл, т. е. будут выполняться все операции по обработке файла;

mbct — количество буферов, которые должны быть использованы системой FCS при обработке файла, ориентированного на записи.

Если это число больше единицы, то реализуется многоуровневая буферизация;

mbfg — параметр указывает тип многоуровневой буферизации. Может быть указано одно из двух следующих символических значений:

FD.RAH — при обработке файла должна быть использована операция предварительного ввода;

FD.WBH — при многоуровневой буферизации должна быть использована операция довывода;

ovbs — числовое значение, указывающее размер блочного буфера (в байтах) в файловой области памяти FSR, которое отменяет стандартное значение размера блока для некоторых типов устройств;

racc — значением параметра может быть:

FD.RWM — при обработке файла должны быть использованы операции ввода-вывода блоков (READ⊙/WRITE⊙). Если это значение не указано, то предполагается ввод-вывод записей (GET⊙/PUT⊙)

FD.RAN — при обработке файла должен быть использован прямой метод доступа. Если это значение не указано, то предполагается последовательный метод доступа;

FD.PLC — при обработке файла должен быть использован режим указания места. Если это значение не дано, то предполагается режим пересылки;

FD.INS — для последовательных файлов операция PUT⊙ не в конце файла не будет производить его усечение. Если это значение не дано, то усечение будет производиться;

ratt — этот параметр определяет атрибуты записей при создании нового файла. Значением параметра может быть:

FD.CR — вывод записи на терминал или АЦПУ должен предваряться символом перевода строки и завершаться символом возврата каретки;

FD.FTN — первый байт выводимой записи содержит символ управления кареткой, используемый программами на языке Фортран;

FD.PRN — первое слово выводимой записи содержит символ управления кареткой. Если ни одно из значений FD.CR, FD.FTN, FD.PRN не указано, то каждая выводимая на устройство с управлением кареткой запись должна внутри себя иметь управляющие символы;

FD.BLK — записям не разрешается пересекать границы блока. Если это значение не указано, то записи могут пересекать границы блоков;

rsiz — числовое значение, определяющее размер (в байтах) записей фиксированной длины, выводимых в создаваемый файл;

rtyp — этот параметр определяет тип записей в создаваемом файле и может принимать одно из следующих значений:

R.FIX — записи фиксированной длины;

R.VAR — записи переменной длины;

R.SEQ — последовательные записи;

urba — символический адрес пользовательского буфера записей;

urbs — числовое значение, определяющее размер (в байтах) пользовательского буфера записей.

Ряд макровызовов имеет специфические параметры, которые приводятся в соответствующих описаниях.

3.1.2. ФОРМАТЫ МАКРОВЫЗОВОВ НАЧАЛЬНОЙ УСТАНОВКИ БЛОКА FDB ПРИ ТРАНСЛЯЦИИ

Начальная установка полей блока FDB при трансляции требует, чтобы был выдан макровызов FDBDF⊙ для распределения памяти под FDB и для определения начального адреса FDB. Вслед за этим макровызовом могут быть использованы другие макровызовы для установки дополнительной информации в этой структуре.

Размещение блока описания файла FDB (FDBDF⊙). Макровызов распределяет память в программе для блока описания файла. Он должен быть указан в исходной программе один раз для каждого файла, который будет открываться во время выполнения программы. Формат вызова:

```
метка: FDBDF⊙
```

Метка именует конкретный блок FDB, определяет его адрес и используется в исполнительных макровызовах ввода-вывода для ссылки на блок описания файла.

Начальная установка данных в секции атрибутов файла (FDAT⊙A). Макровызов FDAT⊙A используется, когда должен быть создан новый файл. Если файл уже существует, то этот макровызов не нужен, так как секция атрибутов файла будет заполнена информацией из блока заголовка этого файла во время его открытия. Формат вызова:

```
FDAT⊙A rtyp , ratt , rsiz , cntg , aloc
```

Начальная установка данных в секции доступа к записям (FDRC⊙A).

Макровызов FDRC⊙A указывает, какие операции ввода-вывода должны быть использованы при обработке файла: ввод-вывод записей или блоков. Если операции ориентированы на записи, то этот макровызов устанавливает в FDB информацию, необходимую для ввода-вывода записей. Если операции ориентированы на блоки, то дополнительно должен быть указан макровызов FDBK⊙A для установки в FDB информации, необходимой для ввода-вывода блоков. Формат вызова:

```
FDRC⊙A racc , urba , urbs
```

Начальная установка данных в секции доступа к блокам (FDBK⊙A). Макровызов FDBK⊙A используется, ес-

ли обработка файла ориентирована на блоки. Формат вызова:

```
FDBK⊙A bkda , bkds , bkvb , bkef , bkst , bkdu
```

Начальная установка данных в секции открытия файла (FDOP⊙A). Макровызов FDOP⊙A обеспечивает систему FCS полной спецификацией обрабатываемого файла, особенностями доступа и обработки файла. Формат вызова:

```
FDOP⊙A lun , dspt , dfnb , facc , actl
```

Начальная установка данных в секции блочных буферов (FDBF⊙A). Макровызов FDBF⊙A передает системе FCS информацию для управления упаковкой и распаковкой отдельных записей в виртуальном блоке. Этот макровызов требуется только для файлов, ориентированных на обработку записей. Формат вызова:

```
FDBF⊙A efn , ovbs , mbct , mbfg
```

3.1.3. ФОРМАТЫ МАКРОВЫЗОВОВ НАЧАЛЬНОЙ УСТАНОВКИ БЛОКА FDB ПРИ ВЫПОЛНЕНИИ

Блок FDB может быть размещен и установлен во время трансляции программы. Однако содержимое секций блока FDB может быть также установлено или изменено во время выполнения программы путем использования следующих макровызовов, которые устанавливают или изменяют секции: атрибутов файла FDAT⊙R; доступа к записям FDRC⊙R; доступа к блокам FDBK⊙R; открытия файла FDOP⊙R; блочных буферов FDBF⊙R.

Форматы вызовов:

```
FDAT⊙R  fdb , rtyp , ratt , rsiz , cntg , aloc
FDRC⊙R  fdb , racc , urba , urbs
FDBK⊙R  fdb , bkda , bkds , bkvb , bkef , bkst , bkdn
FDOP⊙R  fdb , lun , dspt , dfnb , facc , actl
FDBF⊙R  fdb , efn , ovbs , mbct , mbfg
```

Для параметров макровызовов установки блока FDB при выполнении программы нет значений по умолчанию (кроме fdb). Во время выполнения программы используются текущие значения FDB, если они не отвергаются явно указанными параметрами макровызовов.

3.1.4. ФОРМАТЫ МАКРОВЫЗОВОВ НАЧАЛЬНОЙ УСТАНОВКИ ФАЙЛОВОЙ ОБЛАСТИ ПАМЯТИ FSR

Файловая область памяти FSR есть область памяти, размещенная в пользовательской программе как буферный пул и используемая для выполнения операций ввода-вывода записей.

Эта область содержит также управляющую информацию FCS, используемую при обработке всех файлов, независимо от того, ориентированы они на записи или блоки.

1. Начальная установка области FSR при трансляции (FSRSZ⊙). *Макровызов FSRSZ⊙* выделяет область для блочного буферного пула в именованной программной секции ⊙⊙FSR1 и область для управляющей информации FCS в именованной программной секции ⊙⊙FSR2. Формат вызова:

FSRSZ⊙ fbufs , bufsiz , psect

где fbufs — максимальное число блочных буферов, одновременно используемых всеми открытыми файлами для одновременной обработки ввода-вывода записей;

bufsiz — объем в байтах полного пространства пула блочных буферов. Если этот параметр опущен, объем вычисляется по формуле: fbufs*512;

psect — имя .PSECT (программной секции), в которой указан макровызов FSRSZ⊙ Если параметр опущен, подразумевается неименованная программная секция.

Размер буферного пула может быть изменен во время построения задачи (см. необязательные параметры EXTSCT, ACTFIL программы построения задач TKB)

2. Начальная установка области FSR при выполнении (FINIT⊙). *Макровызов FINIT⊙* необходимо включать в повторно запускаемых программах для первоначальной установки или восстановления области FSR.

Следует отметить, что функции макровызова FINIT⊙ выполняются при открытии любого файла, если к этому моменту область FSR не сформирована. Формат вызова:

метка: FINIT⊙

где метка — необязательный, указанный пользователем символ, на который могут передавать управление другие инструкции при выполнении повторно запускаемой программы.

3.1.5. ФОРМАТ МАКРОВЫЗОВА ДЛЯ СОЗДАНИЯ БЛОКА ИМЕНИ ФАЙЛА ПО УМОЛЧАНИЮ

Блок имени файла по умолчанию — средство, обеспечивающее наряду с блоком описания набора данных систему FCS необходимой информацией о спецификации обрабатываемого файла. Формат вызова:

```
метка: NMBLK⊙ fnam , ftyp , fver , dvnm , unit
```

где метка — идентификатор, определяющий адрес блока имени файла по умолчанию для ссылок на него;

fnam — имя файла по умолчанию (от 1 до 9 знаков);

ftyp — тип файла по умолчанию (от 1 до 3 знаков);

fver — номер версии файла по умолчанию;

dvnm, unit — имя и номер устройства по умолчанию, на котором установлен том с обрабатываемым файлом.

Информация из блока имени файла по умолчанию используется только в том случае, если она указана в макровызове NMBLK⊙, но отсутствует в блоке описания набора данных.

3.1.6. ФОРМАТЫ МАКРОВЫЗОВОВ ОТКРЫТИЯ ФАЙЛОВ

Прежде чем какой-либо файл будет обрабатываться пользовательской или системной программой, он должен быть открыт. К моменту открытия в блоке описания файла FDB и вспомогательных блоках должна присутствовать вся необходимая информация (режим и вид обработки файла, доступ к файлу, спецификация файла и т. д.). Любой файл может быть открыт при помощи *универсального макровызова* OPEN⊙. Формат вызова:

```
OPEN⊙ fdb , facc , lun , dspt , dfnb , racc , urba , urbs , err
```

Тип доступа к файлу определяется параметром facc макровызова OPEN⊙. Наряду с этим универсальным макровызовом пользователю предоставляется ряд дополнительных макровызовов для открытия файлов с конкретным типом доступа.

Обработка, которой пользователь намеревается подвергнуть файл, указывается путем буквенного приращения к названию макровызова. Допустимы следующие приращения:

R — читать существующий файл;

W — писать (создать) новый файл;

M — корректировать существующий файл без изменения его длины;

U — корректировать существующий файл и расширять его, если необходимо;

A — добавлять данные в конец существующего файла.

Формат общего макровызова открытия файла.

OPEN⊙x fdb, lun, dspt, racc, urba, urbs, err

где *x* — буквенное приращение к названию макровызова.

Формат макровызова открытия файла для разделенного доступа:

OPNS⊙x fdb, lun, dspt, racc, urba, urbs, err

где *x* — буквенное приращение к названию макровызова

Формат макровызова создания и открытия временного файла:

OPNT⊙W fdb, lun, dspt, racc, urba, urbs, err

Макровызов OPNT⊙W отличается от OPEN⊙W тем, что создаваемый файл не включается ни в один пользовательский каталог файлов. При использовании макровызова OPNT⊙W пользователь несет ответственность за подготовку вре́менного файла к удалению. Формат макровызова создания, открытия и подготовки к удалению временного файла:

OPNT⊙D fdb, lun, dspt, racc, urba, urbs, err

Вре́менный файл, открытый макровызовом OPNT⊙D, автоматически удаляется в момент его закрытия.

Формат макровызова открытия файла по его идентификатору:

OFID⊙x fdb, lun, dspt, racc, urba, urbs, err

где *x* — буквенное приращение к названию макровызова.

Макровызов OFID⊙x выдается для открытия существующего файла с использованием информации, хранящейся в поле идентификатора файла (поле N.FID) блока имени файла. Обычно макровызов OFID⊙x используется совместно с подпрограммами .PARSE и .FIND.

Формат макровызова открытия файла по блоку имени файла:

OFNB⊙x fdb, lun, dspt, racc, urba, urbs, err

где *x* — буквенное приращение к названию макровызова.

Обычно *макровызов OFNB⊙x* используется совместно с подпрограммой .PARSE или с макровызовом NMBLK⊙ и подпрограммами .GTDIR, .GTDID. Если в блоке имени файла поле идентификатора файла (поле N.FID) содер-

жит ненулевое значение, то макровызов OFNB⊙x исполняется так же, как OFID⊙x.

Макровызовы OFID⊙x и OFNB⊙x целесообразно применять в программах с перекрытиями, используя обращения к подпрограммам в одной ветви дерева перекрытий, а макровызовы — в другой.

3.1.7. ФОРМАТЫ МАКРОВЫЗОВОВ ОБРАБОТКИ ЗАПИСЕЙ

При работе с файлом, ориентированным на обработку записей, могут быть использованы две операции — читать и (или) писать логическую запись.

1. Читать логическую запись (GET⊙). *Макровызов GET⊙* считывает логическую запись из файла и помещает в FDB адрес (в поле F.NRBD+2) и длину в байтах (в поле F.NRBD) считанной записи. В режиме пересылки всегда, а в режиме указания места для переходящих записей считываемая логическая запись перемещается из блочного буфера области FSR в пользовательский буфер записей, адрес и длина которого должна быть сформирована пользователем в полях F.URBD+2 и F.URBD блока описания файла FDB соответственно.

Если считывание производится в режиме прямого доступа, то номер читаемой записи должен быть предварительно помещен в поля F.RCNM и F.RCNM+2 соответствующего FDB. Это значение при открытии файла устанавливается в единицу, а после каждой операции чтения увеличивается на единицу.

В режиме прямого доступа могут обрабатываться только записи фиксированной длины, а в режиме последовательного доступа — записи переменной длины и последовательные записи. Если считанная запись была последовательной, то независимо от режимов обработки файла последовательный номер помещается в поле F.SEQN. Формат вызова:

```
GET⊙ fdb, urba, urbs, err
```

2. Читать логическую запись в режиме последовательного доступа (GET⊙S). *Макровызов GET⊙S* может быть использован только для файлов, открытых для обработки в режиме последовательного доступа. Формат вызова:

```
GET⊙S fdb, urba, urbs, err
```

3. Читать логическую запись в режиме прямого доступа (GET⊙R). *Макровызов GET⊙R* позволяет в отличие от макровызова GET⊙ указать в качестве параметра номер считываемой записи. В остальном оба макровызова идентичны. Формат вызова:

GET⊙R fdb, urba, urbs, lrcnm, hrcnm, err

где lrcnm — младшие 16 бит номера считываемой записи (помещается в поле F.RCNM+2 блока FDB);

hrcnm — старшие 15 бит номера считываемой записи (помещается в поле F.RCNM блока FDB).

4. Писать логическую запись (PUT⊙). *Макровызов PUT⊙* используется для вывода логической записи в файл. Для операции PUT⊙ поля F.NRBD+2 и F.NRBD в соответствующем FDB должны содержать адрес и размер (в байтах) выводимой записи. В режиме пересылки пользователь может создавать записи в произвольной области своей программы, но не в пользовательском буфере записей, определенном в FDB (в полях F.URBD+2 и F.URBD).

В режиме указания места пользователь должен создавать выводимую запись по адресу, определенному системой в поле F.NRBD+2, и в макровызове PUT⊙ необходимо указать только размер записи, который будет занесен в поле F.NRBD. В поле F.NRBD+2 система заносит адрес внутри блочного буфера области FSR или адрес пользовательского буфера записей F.URBD+2, если остаток блочного буфера недостаточен для размещения записи максимальной длины, которая определяется длиной пользовательского буфера (в поле F.URBD).

Если записи в файле являются последовательными записями, то поле F.SEQN соответствующего FDB содержит последовательный номер, который должен изменяться пользователем.

Операция PUT⊙ прямого доступа требует указания номера выводимой записи в полях F.RCNM и F.RCNM+2 соответствующего FDB. Это значение при открытии файла устанавливается в единицу, а после каждой операции увеличивается на единицу. Операции PUT⊙ прямого доступа в режиме указания места требуют использования подпрограммы .POSRC. Формат вызова:

PUT⊙ fdb, nrba, nrbs, err

где nrba — адрес области памяти, в которой подготовлена для вывода в режиме пересылки очередная запись (помещается в поле F.NRBD+2 блока FDB);

nrbs — длина (в байтах) выводимой записи переменной длины или последовательной записи (помещается в поле F.NRBD блока FDB).

5. Писать логическую запись в режиме последовательного доступа (PUT⊙S). *Макровызов PUT⊙S* может быть использован только для файлов, открытых для обработки в режиме последовательного доступа. Формат вызова:

PUT⊙S fdb, nrba, nrbs, err

Макровызов PUT⊙S имеет тот же формат и те же параметры, что и PUT⊙.

6. Писать логическую запись в режиме прямого доступа (PUT⊙R). *Макровызов PUT⊙R* позволяет в отличие от макровызова PUT⊙ указать в качестве параметра номер выводимой записи. В остальном оба макровызова идентичны. Формат вызова:

PUT⊙R fdb, nrba, nrbs, lrcnm, hrcnm, err

где lrcnm, hrcnm — младшая и старшая части номера выводимой записи (помещаются в поля F.RCNM+2 и F.RCNM блока FDB соответственно).

3.1.8. ФОРМАТЫ МАКРОВЫЗОВОВ ОБРАБОТКИ БЛОКОВ

При работе с файлом, ориентированным на обработку виртуальных блоков, могут быть использованы операции чтения и записи блоков, а также операция ожидания завершения ввода-вывода.

1. Читать виртуальный блок (READ⊙). *Макровызов READ⊙* обеспечивает ввод блока данных с устройства, например с диска, и, если указаны значения необязательных параметров, возврат в блоке состояния ввода-вывода информации о завершении операции и передачу управления подпрограмме обслуживания асинхронного системного прерывания. Формат вызова:

READ⊙ fdb, bkda, bkbs, bkvb, bkef, bkst, bkdn, err

2. Писать виртуальный блок (WRITE⊙) *Макровызов WRITE⊙* обеспечивает вывод блока данных в файл на устройство, ориентированное на блоки, например диск. Как и для макровызова READ⊙, если указаны значения необязательных параметров, то информация о состоянии передается в блок состояния ввода-вывода, и после завершения операции управление передается обслуживающей подпрограмме асинхронных системных прерываний, которая обеспечивает синхронизацию операций ввода-вывода блоков с выполнением программы пользователя. Формат вызова:

WRITE⊙ fdb, bkda, bkbs, bkvb, bkef, bkst, bkdn, err

3. Ждать завершения ввода-вывода блоков (WAIT⊙). *Макровызов WAIT⊙* используется только совместно с READ⊙ и WRITE⊙ для приостанова выполняемой программы до завершения затребованного ввода-вывода блоков. Формат вызова:

```
WAIT⊙ fdb, bkef, bkst, err
```

3.1.9. ФОРМАТЫ МАКРОВЫЗОВОВ ЗАКРЫТИЯ И УДАЛЕНИЯ ФАЙЛОВ

Ниже перечисляются форматы макровызовов закрытия и удаления файлов и даются краткие характеристики их использования.

1. Закрыть указанный файл (CLOSE⊙). *Макровызов CLOSE⊙* используется для завершения обработки файла. Формат вызова:

```
CLOSE⊙ fdb, err
```

2. Удалить указанный файл (DELET⊙). *Макровызов DELET⊙* используется для удаления файлов пользователя из каталога и освобождения пространства на дисковом томе. Если макровызов применяется для открытого файла, то файл закрывается и удаляется, если для закрытого — удаляется при наличии в спецификации файла в блоке описания набора данных или блоке имени файла по умолчанию номера версии файла. Формат вызова:

```
DELET⊙ fdb, err
```

3.2. ПОДПРОГРАММЫ УПРАВЛЕНИЯ ФАЙЛАМИ

Для вызова подпрограмм управления файлами используется инструкция макроассемблера CALL. Во время построения задачи эти подпрограммы включаются в задачу пользователя из системной библиотеки объектных модулей.

Перед вызовом конкретной подпрограммы требуется, чтобы в указанных регистрах была установлена необходимая информация. Эти требования даны в описаниях соответствующих подпрограмм.

Как правило, если подпрограммой управления файлами обнаруживается ошибка, то устанавливается бит C в слове состояния процессора и код ошибки устанавливается в поле F.ERR блока описания файла FDB. Однако неко-

торые подпрограммы в силу специфического характера своих функций ошибок не фиксируют. К ним относятся подпрограммы .RDFDR, .WDFDR, .RDFUI, .WDFUI, .RDFPP, .WDFPP.

Ниже перечисляются подпрограммы управления файлами и дается краткое описание их использования.

1. Подпрограммы .RDFDR и .WDFDR предназначены для чтения и записи описателя символьной строки кода идентификации каталога по умолчанию, который используется при открытии файла, если соответствующей информации нет в спецификации файла (ни в блоке описания набора данных, ни в блоке имени файла по умолчанию).

Подпрограмма .WDFDR заносит описатель строки каталога по умолчанию в секцию ⊙⊙FSR2 файловой области памяти FSR. Входные параметры: R1 — размер в байтах новой строки, R2 — адрес строки.

Подпрограмма .RDFDR читает описатель строки каталога по умолчанию из программной секции ⊙⊙FSR2. Выходные параметры: R1 — длина строки в байтах; R2 — адрес строки. Если описатель строки каталога не был предварительно занесен в ⊙⊙FSR2 подпрограммой .WDFDR, R2 содержит нуль, и система FCS в качестве идентификатора каталога по умолчанию использует код UIC по умолчанию.

2. Подпрограммы .RDFUI и .WDFUI предназначены для чтения и записи кода UIC по умолчанию, который хранится в двоичном виде в программной секции ⊙⊙FSR2 (в отличие от символьной строки каталога по умолчанию, описатель которой также хранится в секции ⊙⊙FSR2).

Код UIC по умолчанию используется для определения каталога открываемого файла, если этой информации нет в других источниках. Подпрограмма .WDFUI записывает код UIC по умолчанию в двоичном виде из регистра R1, а подпрограмма .RDFUI читает UIC в регистр R1 (старший байт — номер группы, младший — номер члена).

3. Подпрограммы .RDFFP и .WDFFP предназначены для чтения и записи слова доступа к файлу, которое хранится в программной секции ⊙⊙FSR2. Это слово используется во время создания нового файла для установки кода доступа в блоке заголовка файла. Если значение слова доступа равно —1, то при создании файла будет использован код доступа по умолчанию, установленный для тома ключом FPRO команд оператора INITVOLUME или MOUNT. Подпрограмма .RDFFP читает слово

доступа к файлу в регистр R1, а подпрограмма .WDFFP записывает слово доступа к файлу из регистра R1.

4. Подпрограммы .RFOWN и .WFOWN предназначены для чтения и записи идентификатора владельца файла, который хранится в программной секции ⊙⊙FSR2. Этот идентификатор заносится в блок заголовка вновь создаваемого файла и используется совместно с кодом доступа к файлу для его сохранности. Если идентификатор владельца в секции ⊙⊙FSR2 равен нулю, то при создании файла будет в качестве кода владельца использован код UIC задачи. Подпрограмма .WFOWN записывает идентификатор владельца в двоичном коде из регистра R1, а подпрограмма .RFOWN читает его в регистр R1 (старший байт — номер группы, младший — номер члена).

5. Программа .ASCPP преобразует символьную строку каталога в двоичный код UIC. Входные параметры: R2 — адрес двухсловного описателя строки каталога (первое слово — длина, второе слово — адрес строки), R3 — адрес слова в пользовательской программе, в которое должен быть возвращен двоичный код UIC (старший байт — номер группы, младший — номер члена).

Подпрограмма .ASCPP преобразует двоичный код UIC, содержащийся в регистре R3, в символьную строку каталога. Регистр R2 должен содержать адрес области памяти, в которую будет помещена строка каталога «[n, m]», регистр R4 — управляющие биты (бит 0=0, если нужно подавлять ведущие нули, бит 1=0, если в строку каталога надо поместить ограничительные квадратные скобки и разделительную запятую). В результате работы подпрограммы регистр R2 будет содержать адрес первого байта после строки каталога.

6. Подпрограммы .PARSE, .PRSDV, .PRSDI, .PRSFN и .ASLUN выполняют функции, относящиеся к указанному блоку имени файла.

Подпрограмма .PARSE сначала очищает блок имени файла, а затем заполняет его информацией об имени и двоичном номере устройства, идентификаторе каталога, имени, типе и версии файла. Кроме того, она вызывает подпрограмму .ASLUN для назначения логического номера устройства. Подпрограмма .PARSE предполагает, что файловая область FSR задачи установлена (явно макровызовом FINIT⊙ или неявно макровызовом OPEN⊙). Подпрограмма .PARSE использует информацию из программной секции ⊙⊙FSRR2, блока описания набора данных и блока имени файла по умолчанию. Вход-

ные параметры: R0 — адрес FDB, R1 — адрес блока имени файла (необязательно в том же FDB), R2 — адрес блока описания набора данных (или нуль, если блок не должен использоваться), R3 — адрес блока имени файла по умолчанию (или нуль, если блок не должен использоваться). Адреса в регистрах R2 и R3 не обязаны совпадать с адресами соответствующих блоков, указанных в FDB.

Подпрограммы .PRSDV, .RRSDI, .PRSFN отличаются от .PARSE тем, что формируют в предварительно обнуленном блоке имени файла информацию только об устройстве, каталоге или имени файла соответственно.

Подпрограмма .ASLUN назначает логический номер указанному устройству и заполняет соответствующие поля в FDB и блоке имени файла. Входные параметры: R0 — адрес FDB, R1 — адрес блока имени файла (не обязательно в том же FDB). Если устройство не указано, то берется устройство, предварительно назначенное данному логическому номеру. Если же логический номер предварительно не назначен, то он назначается устройству SY:.

7. Подпрограммы .FIND, .ENTER и .REMOV предназначаются для поиска, включения и удаления записей в каталогах. Термин «запись каталога» относится к каталогам пользовательских файлов UFD и к главному каталогу MFD.

Подпрограмма .FIND просматривает файл каталога, указанный идентификатором в блоке имени файла (поле N.DID) и заполняет поле идентификатора файла N.FID в блоке имени файла (структура блока имени файла приведена в приложении). Входные параметры: R0 — адрес блока FDB, R1 — адрес блока имени файла (не обязательно в том же FDB). Подпрограмма допускает значения 0 и —1 в качестве наибольшего и наименьшего номера версии файла в каталоге, просматривает каталог, отыскивает в нем нужную запись и возвращает в блок имени файла действительный номер версии файла. В имени файла могут присутствовать символы «*», что отмечается битами в поле N.STAT. В этом случае подпрограмма .FIND просматривает каталог, отыскивая первую запись, удовлетворяющую требуемой информации, формирует в блоке имени файла действительные имя, тип и версию файла, а также поле N.NEXT для последующих поисков.

Последовательная обработка файлов, спецификация которых содержит символ «*» (но не в строке каталога), может выполняться следующим образом:

вызов подпрограммы .PARSE, которая, в частности, обнуляет поле N.NEXT и заполняет поле N.STAT;

проверка в поле N.STAT наличия символа «*» в спецификации файла;

если символа «*» нет, файл обрабатывается обычным образом, если «*» есть, то вызывается подпрограмма .FIND для получения в блоке имени файла явной спецификации;

если получена ошибка IE.NSF, то в каталоге отсутствует очередная запись, удовлетворяющая спецификации с символом «*»;

если подпрограмма .FIND завершилась без ошибок, то файл открывается, обрабатывается, затем снова вызывается подпрограмма .FIND и т. д.

Спецификация файла, допускаемая подпрограммой .FIND, должна удовлетворять следующему ограничению: если символ «*» присутствует в поле имени или типа файла, версия не может иметь значения 0 или —1. Для удаления файла, спецификация которого содержит знак «*», нужно сохранить поля N.STAT и N.NEXT, обнулить их, удалить файл и восстановить значения этих полей для продолжения работы.

Входные данные для подпрограмм .ENTER и .REMOV: R0 — адрес блока FDB, R1 — адрес блока имени файла (не обязательно в том же FDB). Если при включении записи в каталог номер версии равен нулю, то формируется запись с номером версии, на единицу большую максимально имеющейся. Окончательный номер версии возвращается в блок имени файла. При включении символ «*» не допускается. Подпрограмма .REMOV допускает символ «*» по правилам подпрограммы .FIND, но не допускает 0 и —1 в качестве номера версии.

8. Подпрограммы .GTDIR и .GTDID предназначены для включения в трехсловное поле N.DID блока имени файла идентификатора каталога. Подпрограмма .GTDIR берет информацию о каталоге из двухсловного описателя строки каталога, адрес которого должен содержаться в регистре R2. Этот описатель строки определяет размер и адрес строки каталога. Подпрограмма .GTDID для поиска каталога использует двоичное значение UIC по умолчанию в программной секции ⊙⊙FSR2 файловой области FSR. Кроме того, входными параметрами для обеих подпрограмм являются: R0 — адрес FDB, R1 — адрес блока имени файла (не обязательно в том же FDB), в частности адрес блока имени файла по умолчанию (NMBLK⊙).

9. Подпрограммы .POINT, .MARK, .POSRC и .POSIT предназначены для определения и изменения текущей позиции внутри файла. Подпрограмма .MARK сохраняет информацию о текущей позиции в файле. Входные параметры: R0 — адрес FDB; выходные параметры: R1 и R2 — старшая и младшая части виртуального номера блока, R3 — номер следующего байта в блоке. Сохранив эту информацию, можно, когда потребуется, снова открыть файл и воспользоваться подпрограммой .POINT, которая по исходным данным из регистров R0 — R3 восстанавливает позиционирование файла. В режиме указания места подпрограмма .POINT устанавливает поле F.NRBD+2 для выполнения последующей операции PUT⊙. Таким образом, при помощи подпрограмм .MARK и .POINT можно имитировать режим прямого доступа для последовательных файлов с записями переменной длины.

Подпрограмма .POSRC определяет местоположение записи фиксированной длины внутри файла, используя значения полей F.RSIZ, F.RCNM и F.RCNM+2 блока FDB, адрес которого должен находиться в регистре R0. В режиме указания места подпрограмма устанавливает в поле F.NRBD+2 адрес, по которому пользователь должен построить новую запись перед выполнением операции PUT⊙ при прямом методе доступа.

Подпрограмма .POSIT по номеру и длине записи фиксированной длины возвращает информацию о ее местоположении внутри файла, аналогичную информации подпрограммы .MARK. В отличие от .POSRC подпрограмма .POSIT не выполняет позиционирование в файле, а только вычисляет информацию о позиционировании для последующей операции .POINT.

10. Описываемая далее группа подпрограмм предназначена для выполнения определенного действия по отношению к файлу.

Подпрограмма .RENAM переименовывает файл в указанном каталоге (открытый файл предварительно закрывается). Входные параметры: R0 — адрес FDB, связанного с исходным именем файла; R1 — адрес FDB, содержащего информацию о новом имени, логическом номере устройства и флаге событий.

Подпрограмма .TRNCL используется для усечения дискового файла, т. е. освобождения всех блоков в конце файла, выделенных для него, но не заполненных данными. После операции усечения последний блок файла будет содержать последнюю запись (точку EOF). Входной па-

раметр: R0 — адрес FDB. Усекаемый файл должен быть открыт с правами записи и расширения, после усечения файл закрывается.

Подпрограмма .EXTND используется для расширения файлов. Входные параметры: R0 — адрес FDB; R1 — младшие 16 разрядов числа, указывающего количество блоков расширения; R2 (старший байт) — старшие 8 разрядов числа, указывающего количество блоков расширения; R2 (младший байт) — управляющий байт со следующими возможными значениями:

200 — расширение должно быть прерывистым (очищается признак непрерывности файла);

201 — к файлу должна быть добавлена непрерывная область;

203 — к файлу должна быть добавлена наибольшая имеющаяся непрерывная область, если нет в наличии непрерывной требуемой длины (очищается признак непрерывности файла);

205 — первоначальное расширение, файл должен быть непрерывным;

207 — к файлу должна быть добавлена наибольшая имеющаяся непрерывная область, если нет в наличии требуемой (файл остается непрерывным);

210 — прерывистое расширение на число блоков, являющееся умолчанием расширения файла на томе (ключ /EXT команд оператора INITVOLUME и MOUNT).

211 — непрерывное расширение прерывистого файла на число блоков, являющееся умолчанием расширения файла на томе (ключ /EXT команд оператора INITVOLUME и MOUNT).

Подпрограмма .MRKDL помечает открытый временный файл (созданный, например, макровызовом OPNT⊙W) для удаления. Файл, отмеченный этой подпрограммой, автоматически удаляется не только при закрытии, но и в случае аварийного завершения вызывающей задачи без закрытия файла. Входной параметр: R0 — адрес FDB.

Подпрограмма .DLFNB выполняет действия, аналогичные макровызову DELET⊙: закрывает файл, если он был открыт, и удаляет его. Входной параметр: R0 — адрес FDB. Блок имени файла в FDB должен содержать необходимую информацию.

Подпрограмма .CTRL вызывается для выполнения специальных функций устройства с файлом, адрес FDB которого должен содержаться в R0. Код функции должен быть помещен в R1:

FF.RWD — перемотать набор томов магнитной ленты;

FF.POE — позиционировать к логическому концу набора томов магнитной ленты;

FF.NV — закрыть текущий том магнитной ленты и продолжить операции над файлом на следующем томе;

FF.RWF — возвратиться к началу файла;

FF.SPC — пропустить вперед или назад несколько записей, число которых должно быть предварительно помещено в регистр R2 (положительное число — вперед, отрицательное — назад).

Если подпрограмма .CTRL выполняется для файла на терминале, то происходит очистка признака конца файла на этом терминале.

Перед вызовом подпрограммы .CTRL регистр R3 должен всегда содержать ноль.

3.3. ОБРАБОТКА КОМАНДНЫХ СТРОК

Операционная система обеспечивает возможность динамической обработки командных строк. Для реализации этой возможности пользовательская программа должна быть скомпонована с набором специальных подпрограмм из системной библиотеки объектных модулей.

Этот набор включает подпрограмму ввода командных строк с терминала или из косвенного файла на внешнем носителе (GCML) и подпрограмму интерпретации командных строк (CSI), которая осуществляет синтаксический анализ введенной командной строки и заполняет поля блока описания набора данных, используемого в дальнейшем для открытия файла.

Обычно эти подпрограммы используются совместно, однако каждая из них может быть использована независимо от другой.

Обращение к подпрограммам обработки командных строк осуществляется с помощью макровызовов и требует начальной установки полей управляющих блоков и резервирования общих областей памяти для этих подпрограмм. Макровызовы GCML и CSI используют те же самые соглашения, что и система FCS (автоматически сохраняются все регистры, кроме R0, который содержит адрес соответствующего управляющего блока).

3.3.1. ПОДПРОГРАММА ВВОДА КОМАНДНОЙ СТРОКИ

Подпрограмма ввода командной строки GCML требует наличия управляющего блока командной строки, который создается *макровызовом GCMLB⊙*. Этот макровызов выполняет следующие действия:

резервирует память под управляющие блоки и производит начальную установку его полей;

создает и инициирует блок FDB, используемый для открытия косвенного командного файла на устройстве с файловой структурой или терминале;

создает и инициирует блок имени косвенного командного файла по умолчанию (CMI.CMD);

определяет символические смещения полей управляющего блока и символические значения битов некоторых из этих полей.

Формат вызова:

метка: GCMLB⊙ maxd, prmpt, ubuf, lun, pdl, size

где метка — идентификатор, определяющий адрес управляющего блока командной строки для ссылок на него;

maxd — максимальная глубина вложения косвенных командных файлов, по умолчанию 0;

prmpt — трехсимвольная подсказка по умолчанию, которая выводится на терминал перед вводом командной строки. Символы ⟨CR⟩, ⟨LF⟩, три символа подсказки и символ «> » заносятся в 6-байтное поле G.DPRM управляющего блока. Если параметр опущен, используются три пробела;

ubuf — адрес буфера для ввода строки. Если параметр опущен, буфер выделяется в управляющем блоке;

lun — логический номер устройства, с которого будут вводиться командные строки (обычно этот логический номер назначается пользователем на терминал), по умолчанию 1;

pdl — адрес области для таблицы вложенных косвенных файлов. Если этот параметр задан, то пользователь должен зарезервировать по этому адресу область длиной G.LPDL=16*(maxd+1) байт. Если параметр опущен, то область необходимой длины выделяется внутри управляющего блока;

size — размер в байтах буфера для ввода командной строки, включая два дополнительных байта; по умолчанию 82.

Макровызов GCMLB⊙ устанавливает при трансляции значения по умолчанию в байт состояния и управления (поле G.MODE управляющего блока), которые определяют действия подпрограммы GCML при вводе командной строки. Во время выполнения пользователь может изменить значения битов состояния и управления, которые имеют следующие символические имена:

GE.COM — если бит установлен, то строка, начинающаяся символом «;», рассматривается как комментарий и не передается вызывающей

программе; если бит сброшен, то строки комментариев передаются программе;

GE.IND — если бит установлен, то строка, начинающаяся символом «@», рассматривается как спецификация вложенного косвенного командного файла, из которого надо вводить очередную командную строку; если бит сброшен, то такая строка передается вызывающей программе;

GE.CLO — если бит установлен, то косвенный командный файл временно закрывается после ввода каждой командной строки; если бит сброшен, то текущий командный файл остается открытым на все время его обработки;

GE.CON — если бит установлен, то можно вводить команды, состоящие из нескольких строк; признаком того, что имеется строка продолжения, является символ «—» в конце текущей строки; установка этого бита возможна только при длине буфера более 82 байт; если бит сброшен, то строки продолжения не объединяются, а передаются вызывающей программе по отдельности;

GE.LC — если бит установлен, то русские буквы не преобразуются в латинские; если бит сброшен — преобразуются.

После трансляции биты GE.COM, GE.IND, GE.CLO установлены, бит GE.LC сброшен, а бит GE.CON установлен, если длина буфера больше 82.

Ввод командной строки осуществляется с помощью *макровызова GCML⊙* следующего формата:

GCML⊙ gclblk adpr, lnpr

где gclblk — адрес управляющего блока командной строки;

adpr, lnpr — адрес и длина альтернативной подсказки, выводимой на терминал перед вводом командной строки; если эти параметры опущены, то выводится указанная в макровызове GCMLB⊙ подсказка по умолчанию.

Макровызов GCML⊙ помещает описатель введенной командной строки (длину и адрес) в поля G.CMLD и G.CMLD+2 управляющего блока.

При первом обращении к подпрограмме GCML делается попытка получить командную строку, переданную с помощью программы связи с оператором (MCR). Если эта попытка неудачна или если из MCR передана пустая строка, то по блоку описания файла FDB внутри управляющего блока подпрограммы GCML открывается файл для ввода командной строки. Если файл открывается на терминале, то перед вводом строки выводится подсказка.

Пользовательская задача может иметь доступ к блоку FDB внутри управляющего блока GCML. Например, для определения того, с терминала или нет была введена командная строка, можно проверить бит FD.TTY поля F.RCTL блока FDB.

Если требуется закрыть текущий коевенный командный файл и вернуться к первому (не вложенному) файлу, надо выполнить *макровызов RCML⊙*:

```
RCML⊙ gclblk
```

Если бит GE.CLO в управляющем блоке GCML сброшен, т. е. после ввода командной строки файл остается открытым, то его можно временно (до ввода следующей строки) закрыть *макровызовом CCML⊙*:

```
CCML⊙ gclblk
```

Успешное завершение исполнительных макровызовов GCML⊙, RCML⊙, CCML⊙ сопровождается сбросом бита C слова состояния процессора. Если при вводе командной строки обнаружена ошибка, то бит C устанавливается в единицу, и код ошибки заносится в однобайтное поле G.ERR управляющего блока GCML. Коды ошибок имеют следующие символические значения:

GE.IOR — ошибка устройства при вводе строки,
GE.QPR — ошибка при открытии командного файла;
GE.MDE — превышена максимальная глубина вложения командных файлов;
GE.BIF — синтаксическая ошибка в спецификации косвенного командного файла;
GE.RBG — длина вводимой строки (может быть вместе с продолжениями) больше длины буфера; в буфере содержится уместившаяся часть строки;
GE.EOF — признак конца файла обнаружен в первом (не вложенном) командном файле; при вводе строк с терминала признаком конца файла является ввод символа CTRL/Z.

3.3.2. ИНТЕРПРЕТАЦИЯ КОМАНДНОЙ СТРОКИ

Интерпретатор командной строки CSI позволяет обрабатывать строки следующего вида:

```
список-выводных-файлов=список-вводных-файлов
```

Под файлом понимается его спецификация в следующем виде:

```
устройство:[группа, член]имя. тип; версия
```

За спецификацией может следовать один ключ или несколько ключей. Ключи отделяются друг от друга и от спецификации файла знаком «/». Ключи могут быть положительными (указывается имя ключа), или отрицательными (указывается имя, перед которым ставится знак «—» или префикс «NO»). Список выводных файлов и знак «=» могут отсутствовать.

CSI требует специального управляющего блока интерпретатора командных строк, длина которого (символ C.SIZE) определяется макровызовом CSI⊙. Выделение области памяти для управляющего блока может быть выполнено следующей последовательностью директив в пользовательской программе на макроассемблере:

```
        CSI⊙            ; определение полей управляющего блока
                        ; CSI и значений битов
        .EVEN           ; выравнивание блока на границу слова
метка: .BLKB C.SIZE     ; присвоение имени управляющему блоку
                        ; и резервирование памяти
```

Синтаксический анализ командной строки выполняется по *макровызову CSI⊙1*. Кроме анализа, эта макрокоманда уплотняет командную строку, удаляя из нее незначащие символы (пробелы, знаки горизонтальной табуляции, знаки забоя, нули). Формат макровызова:

```
CSI⊙1 csiblk, buff, len
```

где csiblk — адрес управляющего блока интерпретатора;
buff — адрес анализируемой командной строки;
len — длина в байтах анализируемой командной строки.

Обычно анализируется командная строка, введенная по макровызову GCML⊙, т. е. в качестве параметров buff и len макровызова CSI⊙1 удобно использовать значения полей G.CMLD+2 и G.CMLD управляющего блока подпрограммы ввода командных строк.

Если командная строка содержит знак «=», то в байте состояния запроса командной строки (поле C.STAT управляющего блока CSI) устанавливается бит CS.EQU.

Если при анализе командной строки обнаружена синтаксическая ошибка, то устанавливается бит C слова состояния процессора, а описатель (длина и адрес) ошибочного сегмента строки заносится в поле C.FILD управляющего блока CSI.

Макровызов CSI⊙2 выбирает из обработанной макровызовом CSI⊙1 командной строки очередную спецификацию входного или выходного файла, создает блок описания набора данных для открытия файла, соответствующего выбранной спецификации, а также обрабатывает все ключи и их значения, следующие за спецификацией файла. Формат макровызова:

```
CSI⊙2 csiblk, inpout, swtab
```

где csiblk — адрес управляющего блока интерпретатора;
inpout — определяет выбор очередной спецификации из списка входных (INPUT) или выходных (OUTPUT) файлов;
swtab — адрес таблицы описателей ключей; если параметр опущен, то ключи в спецификации файла не обрабатываются.

Шестисловный блок описания набора данных создается внутри управляющего блока CSI, начиная с поля C.DSDS. Это поле состоит из трех двухсловных описателей (длина и адрес) компонентов спецификации файла: имени устрой-

ства (C.DEVD), каталога (C.DIRD) и имени файла (C.FILD). Кроме того, устанавливаются следующие биты в байте состояния C.STAT в управляющем блоке CSI:

CS.NMF,— если спецификация содержит имя (тип, версию) файла, формируется поле C.FILD; иначе этот бит и оба слова поля нулевые;

CS.DIF,— если спецификация файла содержит идентификатор каталога, формируется поле C.DIRD; иначе этот бит нулевой и поле содержит информацию от последнего запроса CSI⊙2 на спецификацию файла того же типа (входного или выходного), содержащей идентификатор каталога;

CS.DVF,— если спецификация файла содержит имя устройства, формируется поле C.DEVD; иначе этот бит нулевой, и поле содержит информацию от последнего запроса CSI⊙2 на спецификацию файла того же типа, содержащую имя устройства;

CS.WLD, если спецификация файла содержит знак «*»;

CS.MOR, если спецификация файла заканчивается запятой, т. е. в командной строке присутствуют еще спецификации того же типа, что и только что обработанная.

Если во время семантического анализа спецификации файла возникла ошибка, то в слове состояния процессора устанавливается бит «С», и управление возвращается вызывающей программе. Возможные причины ошибок: не описанный в таблице ключей, но присутствующий в спецификации файла ключ; недопустимое значение ключа; с ключом связано больше значений, чем указано в таблице ключей; в командной строке присутствует отрицательный ключ, что, согласно таблице ключей, недопустимо.

Таблица описателей ключей создается на этапе трансляции последовательностью макровызовов CSI⊙SW. *Макровызов CSI⊙ND* ограничивает таблицу описателей ключей.

Макровызов CSI⊙SW создает элемент таблицы ключей для каждого ключа, ожидаемого в спецификациях файлов обрабатываемой командной строки. Первый макровызов CSI⊙SW в последовательности должен иметь метку, значением которой становится адрес таблицы ключей и на которую должен ссылаться соответствующий макровызов CSI⊙2 (параметр swtab). Формат:

CSI⊙SW sw, mk, mkw, csflg, nflg, vtab, compflg

где sw — имя ключа не более шести символов, параметр обязательный;
mk — маска ключа, значение которой устанавливается в первом слове масок (поле C.MKW 1 в управляющем блоке CSI), если данный ключ или его отрицание обнаружен в обрабатываемой спецификации файла; обычно в качестве маски используются числа, двоичное представление которых содержит только один единичный разряд (восьмеричные числа 1, 2, 4, 10, 20 и т. д.); во втором слове масок (поле C.MKW 2 в управ-

ляющем блоке CSI) биты маски ключа устанавливаются, если соответствующий ключ присутствует в обрабатываемой спецификации файла и положителен, или сбрасываются, если ключ отрицателен;

mkw — адрес слова масок пользователя; значение этого слова изменяется в зависимости от значения параметра csflg;

csflg — параметр указывает порядок изменения пользовательской маски и может принимать два значения: SET — биты в пользовательской маске (mkw), соответствующие маске ключа (mk), должны быть установлены, если ключ положителен, и сброшены, если ключ отрицателен; CLEAR — наоборот; если параметр опущен, то предполагается значение SET;

nflg — если этот параметр имеет значение NEG, то допускается использование как положительного, так и отрицательного ключа; если параметр опущен, то разрешен только положительный ключ;

vtab — адрес таблицы описателей значений ключа, которая создается макровызовом CSI⊙SV;

compflg — указывает способ сравнения имени ключа в командной строке с именем, указанным в параметре sw; если параметр имеет значение LONG, то все символы имени помещаются в таблицу и при опознавании должны составить начало имени ключа в командной строке, а остальные символы имени ключа в командной строке игнорируются; если параметр имеет значение EXACT, то все символы имени помещаются в таблицу и должны полностью совпасть с именем ключа в командной строке; если параметр опущен, то в таблицу помещаются только первые два символа имени, которые должны совпасть с первыми двумя символами имени ключа в командной строке, остальные символы игнорируются.

Таблица описателей значений ключа создается на этапе трансляции последовательностью *макровызовов CSI⊙SV*. Макровызов CSI⊙ND должен ограничивать (заканчивать) каждую таблицу описателей значений ключа. Для каждого значения ключа, которое пользователь задает для данного ключа в спецификации файла, в программе должна быть определена соответствующая запись в таблице описателей значений ключа. Формат макровызова:

CSI⊙SV type, adr, len

где type — параметр указывает тип представления значения ключа: ASCII — строка знаков в символьном коде; NUMERIC — число со знаком или без него (по умолчанию или с предшествующим символом «#» — восьмеричное, с точкой в конце — десятичное); OCTAL — восьмеричное число без знака; DECIMAL — десятичное число. Если параметр опущен, то по умолчанию предполагается тип ASCII;

adr — адрес области для помещения результирующего значения ключа;

len — длина области в байтах для значения ключа (для числовых типов 2 или 4).

4

КОМАНДЫ ОПЕРАТОРА

Команды оператора системы — это основной инструмент пользователя, работающего с ОСРВ с терминала. Эти команды интерпретируются и выполняются программой связи с оператором MCR. Команды оператора дают возможность управлять распределением ресурсов системы, запускать задачи и управлять ими, получать оперативную информацию о работе системы. Несколько пользователей могут одновременно работать с системой со своих терминалов. При этом с каждого терминала можно запустить одну или несколько задач. ОСРВ обеспечивает возможность многопользовательской защиты файлов, томов и других системных ресурсов от несанкционированного доступа. Многопользовательская защита — это необязательная функция ОСРВ, выбираемая при генерации системы. При ее использовании для работы с терминалом ОСРВ необходимо предварительно выполнить процедуру регистрации (см. команду HELLO) пользователя в системе.

Непосредственно после генерации системы создается файл учетной информации с помощью системной обслуживающей программы ACNT. Каждый элемент файла учетной информации содержит данные о коде идентификации пользователя UIC, пароле и некоторую другую информацию. В процессе регистрации сравнивается информация, содержащаяся в соответствующем элементе файла учетной информации, с ответами пользователя. При успешной регистрации пользователь получает доступ к ОСРВ и может вводить с терминала командные строки. Вопрос о том, какой программе системы адресованы эти строки, решается при создании или дополнении файла учетной информации. Чаще всего интерпретатором ко-

мандных строк является программа MCR, однако пользователь имеет возможность разработать собственный интерпретатор командных строк CLI или использовать любой существующий, отличный от MCR.

Подчеркнутые сообщения в описании команд выдаются на терминал системы.

В п. 4.2 рассматриваются только команды MCR. Вопросы, связанные с разработкой и применением пользовательского интерпретатора CLI, изложены в п. 4.3.

В процессе диалога пользователя с ОСРВ применяются управляющие символы, вводимые с терминала. Перечень этих символов с кратким описанием приведен в приложении 9.

В п. 4.1 рассмотрены основные понятия, определения и термины, применяемые оператором при работе с MCR.

4.1. ОСНОВНЫЕ ПОНЯТИЯ. УСТРОЙСТВА.

При генерации ОСРВ описываются все подключенные к системе устройства. Для удобства работы каждое устройство имеет единственное в системе *имя,* включающее две буквы и необязательно от 1-й до 3-х восьмеричных цифр. Имя всегда заканчивается двоеточием, например, DK3:. Если указано DK:, то это эквивалентно DK0:. В ОСРВ имена всех стандартных устройств фиксированы, и их нельзя изменить. Такое имя называется *физическим именем устройства,* а само устройство — *физическим устройством.* При рассмотрении команд оператора системы устройства обозначаются Устр: или ddnn; dd: — обозначение не конкретного устройства, а типа устройств, к которому относятся все устройства ddnn:

В ОСРВ, как в любой операционной системе, обеспечивается инвариантность программ по отношению к устройствам ввода-вывода. Поэтому в программах операции ввода-вывода адресованы не физическим устройствам, а к некоторому числу, называемому *логическим номером устройства* (LUN). До начала операции ввода-вывода необходимо обеспечить связь LUN с физическим устройством. Эта связь может быть осуществлена: при построении задачи; во время выполнения задачи и командой MCR REASSIGN после установки задачи.

Псевдоустройства — дополнительный механизм ОСРВ, обеспечивающий ссылки на физические устройства. Это несуществующее устройство, представленное в системе своим именем, допустимым набором функций и описыва-

мое соответствующими таблицами. В ОСРВ применяются пять основных псевдоустройств:

CL:— консольный листинг;

CO:— консольный вывод;

LB:— устройство, содержащее системную библиотеку;

TI:— устройство ввода-вывода с терминала;

SY:— системное устройство, содержащее файлы пользователя.

Пользователь ОСРВ может осуществить назначение LUN на псевдоустройство, не заботясь о том, на какое физическое устройство будет переадресовано командой MCR данное псевдоустройство. Кроме того, система автоматически переадресует TI: на тот терминал, с которого загружена задача, а SY: — на диск, содержащий файлы пользователя.

Логические имена устройств — это еще одно средство системы, которое упрощает и делает более гибкими связи LUN с физическими устройствами.

Эти имена вводятся пользователем. Их синтаксис подчиняется тем же правилам, что и синтаксис имен физических устройств. Логические имена устройств могут совпадать с физическими именами устройств, например, DK1: или BF7:

При использовании логических имен устройств пользователь связывает с помощью операции переназначения номер LUN с логическим именем устройства так же, как ранее связывал LUN с физическим именем устройства. Однако для реализации ввода-вывода такой связи недостаточно, и следует с помощью команды оператора ASSIGN связать логическое имя с физическим именем устройства.

Логические имена устройств являются промежуточным звеном между номером LUN и физическими устройствами. Введение такого звена позволяет уменьшить, например, количество операций переназначения при изменении связей нескольких задач с одним физическим устройством. Все задачи связывают свои номера LUN с одним логическим именем устройства. Связь же этого логического имени с физическим устройством осуществляется одной командой оператора ASSIGN. При изменении физического устройства достаточно также единственной команды ASSIGN для организации связей номера LUN всех задач с новым физическим устройством.

ОСРВ поддерживает три типа назначений:

1) локальные справедливы только для задач, запущенных с того терминала, с которого сделаны назначения;

2) регистрационные выполняются в момент входа в систему и действуют до выхода из системы;

3) глобальные справедливы для задач, запущенных со всех терминалов системы.

При выполнении назначений сначала просматривается таблица имен логических устройств, затем имен физических устройств. Если соответствующее логическое имя устройства будет найдено, поиск заканчивается, и LUN связывается с тем физическим устройством, на которое было назначено логическое имя устройства.

Приоритет различных типов назначений соответствует тому порядку, в котором они перечислены.

Физические устройства в ОСРВ делятся на общие, личные и несвязанные. *Общее устройство* — это устройство, с которым может работать любой пользователь, зарегистрировавшийся в системе после того, как привилегированный пользователь смонтировал его и объявил общим. *Личное устройство* — это устройство, которое выделено для определенного терминала командой оператора ALLOCATE. Доступ к этому устройству задач, запущенных с другого терминала, запрещен. *Несвязанное устройство* не является ни частным, ни общим. Если смонтировать том на несвязанном устройстве, то пользователи других терминалов могут повторно выполнить монтировку и получить доступ к устройству.

4.1.1. ТЕРМИНАЛЫ ОСРВ

Терминалы ОСРВ — это дисплеи различных типов или телетайпы, описанные соответствующим образом при генерации системы. Терминалы могут быть *удаленными* или *локальными*. Каждому терминалу при генерации присваивается физическое имя вида TTnn: (nn — номер терминала), по которому на него можно ссылаться. Для выполнения операций ввода-вывода на терминале, с которого запущена задача, следует назначить соответствующий номер LUN на псевдоустройство TI:.

Терминалы ОСРВ делятся на *привилегированные* и *непривилегированные*. Статус привилегированности всегда присваивается главной консоли и по требованию оператора консоли — любому терминалу. Кроме того, привилегированным может быть объявлен любой терминал, если при входе в систему по команде HELLO указаны соответствующие параметры. При работе с ОСРВ с привилегированного терминала следует учитывать, что система не защищена от действий его оператора, поэтому

требуется особая осторожность при выполнении ряда команд и задач.

Все команды оператора ОСРВ делятся на привилегированные и непривилегированные. Привилегированные команды можно вводить только с привилегированного терминала, остальные — с любого.

Терминалы ОСРВ подразделяются также на *присоединенные* и *неприсоединенные*. Терминал называется присоединенным, если его вводом и выводом управляет задача, не являющаяся программой связи с оператором. Присоединение запрашивается задачей и выполняется ОСРВ, если терминал ранее не был присоединен. В результате присоединения предотвращается доступ других задач к данному терминалу. Для того чтобы обратиться к MCR с присоединенного терминала, следует ввести управляющий символ CTRL/C. После выдачи подсказки (см. п. 4.2) оператор может ввести одну команду, адресованную MCR.

В ОСРВ, кроме рассмотренной классификации терминалов, выделяют еще терминалы, называемые *подчиненными*. В отличие от присоединенных подчиненные терминалы отвергают не только весь незапланированный ввод, но и не воспринимают управляющий символ CTRL/C. Оператор терминала, таким образом, не может обращаться к MCR. Статус подчиненности терминала вводится и отменяется или с помощью команды SET программы связи с оператором, или задачей, обслуживающей терминал с помощью специальной функции ввода-вывода. Подчиненные терминалы используют в системах реального времени, где следует исключить влияние непривилегированного пользователя на ход вычислительного процесса (оператор не может обратиться к MCR). Подчиненные терминалы воспринимают управляющие символы CTRL/O, CTRL/S, CTRL/Q, которые необходимы для управления выводом информации на терминал пользователя.

4.1.2. КОД ИДЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ

Пользователи ОСРВ делятся на группы. Количество групп — от 1 до 377_8. Каждая группа может включать от 1 до 377_8 членов группы. Номер группы (g) и номер члена в группе (m) определяют код идентификации пользователя (UIC). UIC=[g, m].

В ОСРВ без многопользовательской защиты каждому терминалу соответствует единственный код идентификации

пользователя, который определяет код UIC по умолчанию и привилегии доступа к файлам. Код UIC может быть в любое время изменен командой оператора SET.

В системе с многопользовательской защитой каждому терминалу соответствуют два кода UIC: один называется *регистрационным,* второй — UIC *по умолчанию.* Регистрационный код UIC фиксируется при входе в систему (см. команду HELLO программы MCR). Если номер группы регистрационного кода UIC $\leqslant 10$, то терминал автоматически получает статус привилегированности. Оператор привилегированного терминала может в любой момент изменить свой код UIC. Для привилегированного терминала код UIC по умолчанию и регистрационный код UIC совпадают. Оператор непривилегированного терминала не может изменить свой регистрационный код UIC, но может изменить код UIC по умолчанию. Регистрационный код UIC определяет привилегии доступа к файлам задач, запущенных с данного терминала. Код UIC по умолчанию автоматически включается системой в спецификацию файла, если код UIC в ней не был задан явно.

4.1.3. ЗАЩИТА ФАЙЛОВ В ОСРВ

Каждому файлу в ОСРВ соответствует слово, называемое *маской защиты файла.* Маска делится на четыре части, каждая из которых связана с одной из четырех принятых в ОСРВ категорий пользователей: системная; владелец файла; член группы, к которой относится владелец; общая.

Доступ к файлу в ОСРВ может получить только задача. Задача, запущенная с некоторого терминала, относится системой к одной из указанных категорий в зависимости от регистрационного UIC этого терминала*. Если номер группы кода UIC $g \leqslant 10_8$, то привилегии доступа соответствуют системной категории. Если код UIC, под которым запущена задача, и код UIC владельца файла совпадают, то привилегии извлекаются из части маски — «Владелец файла». Если совпадают только номера групп — извлекаются из 3-й категории. В остальных случаях — из 4-й категории.

В момент создания файла каждой из перечисленных категорий будет присвоен набор привилегий доступа. Этот

* С привилегированного терминала можно запустить задачу с любым кодом UIC.

набор может включать до четырех привилегий доступа: чтение (R), запись (W), расширение (E), стирание (D) или RWED. Этот набор может быть сокращен, но только путем удаления отдельных привилегий справа налево, например, RWE или RW или R. Можно запретить доступ к файлу требуемым категориям в любом режиме.

Маска защиты файла может быть изменена стандартными средствами ОСРВ.

4.1.4. СПЕЦИФИКАЦИЯ ФАЙЛОВ В ОСРВ

Формат спецификации файла одинаков для всех ссылок на файл в ОСРВ и имеет следующий вид:

устр:[код-идентификации] имя-файла. тип; версия

где: устр: — это принятое в ОСРВ обозначение физического, логического или псевдоустройства, на котором находится файл;

код-идентификации UIC задается парой чисел в квадратных скобках [g, m] определяющей каталог UFD, в котором имеется ссылка на файл;

имя-файла — строка символов (от 1 до 9), включающая большие латинские буквы или цифры, или их сочетание. Имя файла всегда заканчивается точкой;

тип — строка символов (от 1 до 3), определяющая содержимое файла. Хотя пользователь может ввести собственный тип файла, в ОСРВ приняты трехсимвольные типы файлов, приведенные в приложении 8. Тип файла всегда заканчивается точкой с запятой;

Версия — восьмеричное число (от 1 до 77777), определяющее версию файла. При создании файлу присваивается первая версия. При последующих модификациях номер версии увеличивается на единицу. Если в качестве номера версии указать 0 или не указывать версию совсем, то система ищет самую старшую (новую) версию. Если указать минус 1, то — самую старую версию файла.

4.2. ПРОГРАММА СВЯЗИ С ОПЕРАТОРОМ

4.2.1. ВВОД КОМАНДНОЙ СТРОКИ MCR

Командная строка, адресованная MCR, вводится в ответ на подсказку MCR> или подсказку по умолчанию. Обе эти подсказки указывают на готовность системы принять незапланированный ввод, который будет интерпретироваться MCR. Исключения составляют случаи некорректной работы задач, которые присоединили терминал, назначили AST-подпрограмму по вводу незапланированного символа и не выдали подсказку типа tsk>. В последнем случае незапланированный ввод будет принят задачей. Указанная ситуация легко распознается по реакции системы. Формат командной строки MCR:

имя-команды [параметры] [!комментарий] <признак-завершения-строки>

Имя команды, вводимое с терминала, может быть сокращено до первых трех букв или специально оговоренной при описании трехсимвольной аббревиатуры (исключение составляет единственная команда HELP имя которой вводится полностью).

Параметры, используемые в командных строках MCR, могут быть *ключевыми* или *позиционными*. Ключевые параметры (ключи) содержат обычно символ «/» (косая черта) в качестве первого знака. Далее следует имя ключа и (не обязательно) знак равенства, вслед за которым вводится значение ключа. Например: /PMD или /DRV=dd: или /DENS=32.

Комментарии могут вводиться после имени команды и параметров или начиная с первой позиции строки. В первом случае они отделяются восклицательным знаком; во втором случае вся строка, содержащая только комментарии, начинается точкой с запятой.

Признак завершения строки <ВК> необходим для всех команд, кроме OPEN, где используется еще и управляющий символ <ESC> (см. приложение 9).

4.2.2. ОБРАБОТКА КОМАНДНОЙ СТРОКИ ПРОГРАММОЙ MCR

Из введенной командной строки выделяется имя команды. По имени команды MCR определяет по своим таблицам средства обработки. Часть команд обрабатывается внутренними средствами, в остальных случаях запускаются задачи, каждая из которых обрабатывает одну команду. Для активизации такой задачи осуществляется поиск в STD задачи с именем ...tsk, где tsk — имя команды. Если такая задача существует и не запущена ранее с того же терминала, то она активизируется. В противном случае возвращается сообщение об ошибке. Если задача с указанным именем запущена с другого терминала, MCR активизирует копию этой же задачи с именем вида tskTn, где n — номер терминала, запросившего задачу.

4.2.3. СОГЛАШЕНИЯ ОБ ИМЕНАХ ЗАДАЧ

В системе с многопользовательской защитой могут одновременно быть активны несколько копий одной задачи с именем вида ...xxx. Если запрашивается выполнение более чем одной копии с разных терминалов, имя любой задачи, кроме первой, формируется следующим образом:

tsk Tn

где tsk — трехсимвольное имя задачи;
n — номер терминала, с которого она запущена.

Если в команде оператора необходима ссылка на имя задачи, то пользователь терминала, с которого запущена задача, вводит только трехсимвольное имя (т. е. tsk). Оператор привилегированного терминала должен указать полное имя, если задача запущена не с его терминала.

Если задача запускается в режиме установка-запуск-удаление, то ей присваивается имя вида: TTnn, где nn — номер терминала, с которого задача запущена.

При ссылке на имя задачи с привилегированного терминала следует ввести полное имя; при работе на терминале, с которого запущена задача, имя задачи можно совсем опустить. С терминала в каждый момент времени может быть запущена только одна задача в режиме установка-запуск-удаление.

4.2.4. ОПИСАНИЕ КОМАНД ОПЕРАТОРА СИСТЕМЫ

Для удобства поиска описания требуемых команд они расположены в настоящем разделе в порядке, определяемом латинским алфавитом. После обозначения имени команды и ее аббревиатуры приводятся сведения о ее привилегированности (п), непривилегированности (нп) или возможности использования в обоих режимах с оговоренными ограничениями. Некоторые обозначения, применяемые при описании команд, рассмотрены в предыдущих разделах.

Команда ABORT (*ABO* нп/п) имеет формат:

ABO[RT] имя-задачи [/PMD]

/PMD требует распечатки памяти, занимаемой задачей.

Пример:

ABO PIP завершается задача с именем PIPT3, запущенная с терминала TT3: Она была установлена с именем ...PIP.

Команда ACS (ACS п) имеет формат:

ACS устр:/BLKS=n[.]

/BLKS=n[.] определяет размеры файла выгрузки в 512-байтных блоках.

Пример:

ACS DK3:/BLKS=250.

Выделено пространство размером 250 блоков. Число 250 — десятичное. Если требуется прекратить использование файла выгрузки, необходимо ввести: ACS устр:/BLKS=0. Для того чтобы освобожденное пространство на диске могло быть использовано для других целей, следует ввести:

```
>PIP [0,0] CORING.SYS/TR
```

Команда ACTIVE (ACTIVE (ACT нп)) имеет формат:

ACT [IVE] [{/ALL / /TERM=TTnn:}]

Она позволяет отобразить на терминале имена активных задач, запущенных с данного терминала (параметры опускаются), всех активных задач в системе (/ALL) или задач, запущенных с TTnn: (/TERM=TTnn:).

Команда ALLOCATE (ALL п/п/нп) имеет формат:

ALL[OCATE] dd[nn]:[=*ll*nn:] [/TERM=TTnn:] [/TYPE=устр:]

Она позволяет назначить устройство dd[nn]: в качестве личного для данного терминала или для TTnn: (/TERM=TTnn:). Для использования ключа/TERM необходим привилегированный терминал.

*ll*nn: — логическое имя устройства, которое должно быть назначено на ddnn: (это локальное назначение).

Ключ /TYPE позволяет выделить любое доступное устройство указанного типа. Допустимые типы устройств — DP: и DM:

Команда ALTER (ALT n) имеет формат:

ALT [ER] имя-задачи {/PRI=n / /RPRI=n}

Она позволяет изменить приоритет выполнения установленной задачи (/RPRI) или приоритет установленной задачи и приоритет выполнения одновременно (/PRI) Значение приоритета: $1 \leqslant n \leqslant 250$.

Команда ASSIGN (ASN п/нп) распечатывает, определяет или уничтожает назначение логического устройства.

Обозначения: pp — физическое, логическое или псевдоустройство; nn — номер устройства; ll — логическое имя устройства, tt — терминал.

Непривилегированный формат:

ASN ppnn: =*ll*nn:

Локальное назначение *ll* на pp.ASN=[*ll*nn:] уничтожает назначение устройства *ll* или все локальные назначения, если *ll*nn: не указано, ASN распечатывает все локальные и регистрационные назначения для данного терминала.

Привилегированный формат

ASN ppnn: = *ll*nn: /GBL
ASN=[*ll*nn:] /GBL
ASN /GBL

осуществляет, уничтожает или распечатывает глобальные назначения. Кроме того, при отображении дополнительно выводятся регистрационные и локальные назначения для данного терминала.

ASN ppnn:=*ll*nn:/LOGIN[/TERM=ttnn:]

выполняет регистрационное назначение *ll* на pp для терминала tt, если он задан, или для данного терминала.

ASN=[*ll*nn:] /LOGIN[/TERM=ttnn:]

уничтожает регистрационное назначение *ll*nn: для ttnn: или данного терминала или все регистрационные назначения:

ASN /TERM=ttnn:

распечатывает локальные и регистрационные назначения для ttnn:

ASN=[*ll*nn:] /TERM=ttnn:

уничтожает локальные назначения или назначение *ll*nn: для терминала ttnn:

Команда ATL (ATL нп) выводит на терминал детальную информацию о всех активных задачах в системе или о конкретной задаче, если задано ее имя. Формат команды:

ATL [имя-задачи]

Команда BLOCK (BLK нп/п) блокирует установленную или активную задачу. В результате задача не участвует в конкуренции за ресурсы системы и не будет выполняться. Для снятия указанного состояния следует выдать команду UNBLOCK. Формат команды:

BLK [имя-задачи]

Если имя не задано, используется TTnn.

Команда BOOT (BOO п) позволяет загрузить систему ОСРВ, записанную в виде образа задачи в файле на диске. Формат команды:

BOO[T] [спецификация-файла-образа-системы]

Если спецификация не указана, по умолчанию используется:

LB: [текущий UIC] OCPBCM . SYS; посл. верс.

Команда BROADCAST (BRO нп/п) передает сообщение на один или несколько терминалов системы. Форматы команды:

BRO [ADCAST] TTnn: сообщение,

где TTnn: — адресат;

BRO[ADCAST] @ спецификация-файла

где спецификация косвенного командного файла (см. п. 4.5), содержит параметры команды BRO в любом допустимом формате.

Форматы команд для привилегированного пользователя:

BRO[ADCAST] /ALL: сообщение
BRO[ADCAST] /LOG: сообщение

Сообщение передается всем терминалам (/ALL) или тем, с которых введена команда HELLO (/LOG).

Команда BYE (BYE) служит для выхода из системы с многопользовательской защитой. Формат команды:

BYE

Команда CANCEL (CAN нп/п) — исключить задачу из таймерной очереди — имеет формат:

CAN[CEL] имя-задачи

Команда CLOCK QUEUE (CLQ нп) выводит на терминал информацию о задачах, стоящих в очереди на обслуживание по таймеру. Команда имеет формат:

CLQ[UEUE]

Команда DEALLOCATE (DEA нп/п) отменяет статус устройства как личного. Если устройство не указано, освобождаются все устройства данного пользователя. Непривилегированный пользователь может освободить только собственные устройства. Команда имеет формат:

DEA[LLOCATE] устр:

Команда DEVICES имеет формат:

DEV[ICES] { ddnn: / dd: / /LOG }

Она выводит на терминал информацию о всех устройствах, известных системе, об указанном устройстве, об

устройствах заданного типа или всех терминалах, на которых выполнена регистрация пользователя в ОСРВ (/LOG).

Команда DISMOUNT (DMO нп/п) помечает том на устройстве устр:, предотвращая открытие на нем файлов. Не влияет на ранее открытые файлы. Процесс демонтирования завершается системой после закрытия на томе последнего файла. Команда имеет два формата:

```
DMO устр:[метка-тома] /ключ(и)
DMO/USER /ключ(и)
```

Ключи имеют следующие значения:

/DEV — только для привилегированного пользователя. Осуществляет операцию демонтирования для всех пользователей данного устройства.

/USER демонтирует все тома, монтированные данным пользователем;

/TERM = ttnn: — привилегированный пользователь демонтирует все тома, монтированные с терминала ttnn:;

$$/LOCK=\left\{\begin{matrix}N\\U\\V\end{matrix}\right\}$$

используется только для МЛ. После выполнения команды DMO со значением ключа N лента не разгружается, со значением ключа U — разгружается. Привилегированный пользователь может задать значение V. В результате лента не разгружается и доступна системе.

Команда FIX (FIX п) имеет формат:

```
FIX имя-задачи
```

и позволяет зафиксировать задачу в разделе.

Команда FLAGS (FLA нп/п) позволяет распечатывать, создавать или уничтожать групповые глобальные флаги событий. Непривилегированный пользователь может изменять только флаги своей группы (g), привилегированный — любой группы. Команда имеет формат:

$$FLA[GS][[g]\left\{\left[\begin{matrix}/CRE\\/ELIM\end{matrix}\right\}\right]]$$

/CRE создает флаги событий для группы g (UIC = [g, m]), /ELIM уничтожает групповые глобальные флаги, относящиеся к группе g.

Если параметры команды опущены, MCR распечатывает соответствующие флаги.

Команда HELLO (HEL) регистрирует вход пользователя в систему и имеет форматы

1. HEL[LO] {UIC / имя-пользователя} /пароль
2. HEL[LO]
   ACCOUNT OR NAME
   PASSWORD пароль
3. HEL[LO] {UIC / имя-пользователя}
   PASSWORD: пароль

Информация, вводимая при входе в систему, должна быть заранее занесена в файл учетной информации программой ACNT. Код UIC может задаваться следующим образом:

g, m; [g/m]; [g, m]; g/m.

Если код UIC вводится в форматах с косой чертой, MCR не выводит содержимое файла LOGIN.TXT.

Пример:

> HEL [200/200] /SECRET <ВК>

Команда HELP отображает на терминале содержимое текстовых файлов типа HLP. HELP — единственная команда, которую можно выдать до входа в систему. По умолчанию распечатывается содержимое файла

LB:[1,2] MCR.HLP

Команда HOME (HOM нп/п) позволяет модифицировать ряд полей в собственном блоке тома на диске. После изменения проверяется корректность записи. Действие HOME эквивалентно работе команды INI. Разница в том, что INI создает, а HOME изменяет поля собственного блока тома. Перед выполнением HOME необходимо выполнить

INS ⊙ INI/TASK= ... HOM

Формат команды:

HOM[E] устр: метка-тома/ключ(и)

Ключи имеют следующие значения:

/DENS = {HIGN / LOW}

определяет плотность записи на гибком диске; HIGH — двойная, LOW — обычная;

/EXT = счетчик-блоков определяет, на сколько блоков должен быть увеличен файл, если ранее выделенное пространство исчерпано;

/FPRO = [сист., влад., гр., общ.] задает привилегии доступа, которые устанавливаются по умолчанию при создании нового файла на данном диске. Для каждой категории следует задать полный (RWED) или усеченный справа налево набор привилегий доступа;

/LRU=n n задает количество 512-байтных буферов, которые выделяются в памяти для обработки каталогов;

/MXF=mm определяет количество элементов каталога файлов на томе, включая 5 системных файлов;

/NAME=новая-метка-тома;

/OVR — привилегированный ключ, который позволяет изменить метку, не задавая старую метку в команде;

/OWNER=[g, m] определяет владельца тома, аналогичен ключу/UIC;

/PRO=[сист., влад., гр., общ.] определяет защиту тома. Привилегии доступа в этом случае задаются набором RWCD, где C означает разрешение создавать файлы на томе;

/WIN=k задает число указателей извлечений файла, резервируемых в оперативной памяти при открытии файла;

/VI требует распечатку значений ключей для заданного тома и может применяться только вместе с другим ключом.

Пример:

```
HOM DK3: AAKM/WIN=10./VI
```

Команда INITVOLUME (INI, нп/п) служит для создания тома, отвечающего требованиям файловой системы OCPB. Ранее существовавшая на томе информация разрушается. В системе с многопользовательской защитой INI следует выдавать на личное устройство (устр:). Формат команды:

```
INI[TVOLUME] устр:метка-тома /ключ(и)
```

Ряд ключей команд HOM и INI совпадает, поэтому, если ключ появился в тексте без описания, его следует прочитать в команде HOM.

Ключи имеют следующие значения:

/ACCESS=«символ» определяет право доступа к МЛ. Если задан не пробел, только привилегированный пользователь сможет монтировать том. Допустимые символы A÷Z и специальные знаки. По умолчанию используется пробел.

/BAD=значение определяет обработку дефектных блоков. Значения: [MAN] — список дефектных блоков вводится с терминала. [AUTO] — дефектные блоки выявлены программой BAD. [NOAUTO] — считать все блоки годными. [AUTO, MAN] — использовать информацию программы BAD и дополнительно принять список дефектных блоков с терминала. Если среди значений указывалось MAN, то INI выдает на терминал: INI>
<LBN(S)= За знаком равенства следует ввести

описание дефектных областей в формате m : [n], где m — начальный номер дефектной области, n — длина. Если длины не указаны, то n=1. Несколько дефектных областей вводятся через запятую.

/DENS=$\left\{\begin{matrix}\text{LOW}\\\text{HIGH}\end{matrix}\right\}$/EXT=Сч. блоков, /FRRO=[сист., влад., гр., общ.]

/INDX=размещение-индексного-файла-на-диске

задается для оптимизации доступа. Возможные размещения: BEG — в начале, MID — в середине, END — в конце, BLK:nnn — начиная с указанного номера блока, по умолчанию — в середине.

/INF=n определяет начальное количество заголовков файлов, которое должно быть зарезервировано в индексном файле. 5 системных файлов не входят в это количество.

/LRU=n /MXF=m /OWNER=[g, m]

/OWNER=«owner» используется только для МЛ и позволяет для совместимости с другими системами создать метку тома, содержащую информацию, заданную в кавычках в поле метки — «Владелец».

/PRO=[сист., влад., гр., общ.] /UIC=[g, m] то же, что и /OWNER
/WIN=n /VI

Пример:

```
>INI DK5: ABC/OWNER=[1,7] /FPRO=[RWED,,R]
```

Команда INSTALL (INS n) устанавливает задачу в системе и имеет формат:

INS[TALL] [⊙] спецификация-файла-образа-задачи [/ключ(и)]

При задании ⊙ перед стандартной спецификацией файла система ищет его на устройстве LB: в системном каталоге. Ключи команды INS используются в основном для корректировки параметров задачи, определенных аналогичными ключами при построении задачи (см. описание программы TKB). Ключи команды INS перекрывают действие ключей TKB.

/CKP=$\left\{\begin{matrix}\text{YES}\\\text{NO}\end{matrix}\right\}$ Задача может быть выгружена на диск
Невыгружаемая задача

/CLI=$\left\{\begin{matrix}\text{YES}\\\\\text{NO}\end{matrix}\right\}$ Усганавливаемая задача — интерпретатор командных строк
Не является CLI

/INC=n Ключ позволяет в системах с диспетчером памяти (ДП) увеличить на n слов память, отведенную задаче в системно-управляемом разделе.

/IOP=$\left\{\begin{matrix}\text{YES}\\\text{NO}\end{matrix}\right\}$ Указывает, что привилегированная задача использует границу внешних устройств
Не использует страницу внешних устройств

/PAR=имя-раздела. Позволяет указать имя раздела, где будет выполняться задача. В системах без ДП, база раздела не может быть изменена

/PMD={YES / NO} При аварийном завершении выдается распечатка памяти задачи, если в системе установлена задача PMD...
Не выдается распечатка памяти

/PRI=n Позволяет задать приоритет задачи $1 \leqslant n \leqslant 250$.

/SLV={YES / NO} Устанавливаемая задача является подчиненной.
Не является подчиненной.

/TASK=имя Позволяет установить имя, под которым задача будет запущена

Примеры:

⩾ INS ⊙ PIP

⩾ INS ⊙ MAC /INC=4096

Команда LOAD (LOA n) загружает нерезидентный драйвер. Если задан ключ/PAR, драйвер будет загружен в указанный раздел. Ключ/SIZE задает минимальную длину требуемого раздела (подраздела) в словах ./HIGH позволяет загрузить драйвер в верхнюю часть системно-управляемого раздела. Формат команды:

LOA[D] dd: [ключ(и)]

Формат ключей:

/PAR=имя-раздела
/SIZE=длина-раздела /HIGH.

Команда LUNS (LUN нп) выводит на терминал назначения на физические устройства для указанной в команде задачи. Команда имеет формат:

LUN[S] имя-задачи

Команда MOUNT (MOU нп/п) строит блок управления томом (VCB) для заданного устройства и открывает доступ к тому соответствующему ACP. Команда MOU должна быть выдана перед любым использованием тома в системе (за исключением отдельных случаев). Команда имеет отдельные ключи для дисков и МЛ.

Формат MOU для дисков:

MOU[NT] ddnn: [метка-тома] [/ключ(и)]

Ключи имеют следующие значения.

/ACP=имя; /DENS=плотность; /EXT=сч. блоков; /FOR; /FPRO= [сист, влад, гр., общ.]; /LOCK=значение;
/LRU=n; /OVR; /PAR=«параметры»;
/PRO= [сист, влад, гр., общ.]; /PUB; /[NO]SHARE;

/UIC=[g, m;] /UNL; /VI; /[NO]WAIT; /WIN=n; /[NO]WRITE.

Формат MOU для МЛ:

MOU[NT] список-устройств: метка-тома /ключ(и)

Ключи имеют следующие значения.

/ACP=имя; /BS=n; /CC=парам.; /DENS=плотность;

/FOR; /FPRO=[сист, влад, гр., общ.]; /[NO]HDR3; /[NO]LABEL;

/LOCK=знач.; /OVR; /OVRACC;/OVREXP; /OVRFSID; /PARM=«параметр»; /PRO=[сист, влад, гр, общ.]/PUB /RS=n; /[NO]SHARE; /TR=знач.; /UIC=[g,m]; /VI; /VOL=(список); /[NO]WAIT; /[NO]WRITE.

Ряд ключей служит для модификаций (на время работы с томом в данном сеансе) значений, установленных командой INI и измененных командой HOM. Эти ключи рассмотрены при описании указанных команд и повторно не описываются.

/ACP=имя-задачи-ACP определяет имя задачи, которая должна обслуживать монтируемый том. ОСРВ содержит стандартные ACP для дисков и МЛ, которые выбираются по умолчанию, и обслуживают все устройства данного типа. Для ускорения обработки можно загрузить еще одну копию стандартного ACP. В этом случае вместо имени указывают UNIQUE. После демонтирования тома копия будет удалена.

/BS=n определяет размер блока по умолчанию для МЛ без метки или со стандартными метками: $14 \leqslant n \leqslant 16000$ для лент без метки; $18 \leqslant n \leqslant 16000$ — со стандартными метками.

/CC=парам указывает тип записи на МЛ. Значения: FOR, LIST, NONE определены так же, как в операторе OPEN Фортрана.

/FOR используется для томов, не поддерживающих файловую структуру ОСРВ. Предотвращает доступ других пользователей к данному тому. Если не указано имя ACP, том не доступен файловой системе.

/[NO]HDR3 указывает, следует ли записывать метку HDR3 во вновь создаваемые файлы. Не используется, если система не содержит средств обработки этой метки.

/[NO]LABEL МЛ со стандартными метками (/LABEL) или непомеченная (/NOLABEL); по умолчанию — /LABEL./OVR — ключ для привилегированного терминала (п), позволяет монтировать тома, не зная метки.

/OVRACC (п) отменяет защиту МЛ.

/OVREXP (п) позволяет осуществлять перезапись, несмотря на неистекшую дату хранения.

/OVRFSID используется для нестандартных МЛ, позволяет игнорировать существующий набор идентификаторов файла.

/PARM=«параметры» позволяет передать параметры нестандартному ACP.

/PUB разрешает доступ всех пользователей к данному тому.

/RS=n определяет размер записи на МЛ без меток или со стандартными метками. Значение n: $1 \leqslant n \leqslant m$, где m — размеры блока, заданные в ключе /BS.

/[NO]SHARE/SHARE определяет, что том может быть многократно монтирован пользователями с разных терминалов. Доступ к тому предоставляется в соответствии с привилегиями каждого пользователя. /NOSHARE — том выделен для одного пользователя.

/TR=значение задает способ перекодировки информации при чтении и записи. NONE — нет перекодировки. EBC DIC — перекодировка из ДКОИ в КОИ-7 при чтении и из КОИ-7 в ДКОИ при записи. UT1, UT2, UT3 задают перекодировку по пользовательским таблицам, предварительно подключенным к MTAACP.

/UNL разрешает запись в индексный файл.

/VOL=(список) позволяет перечислить идентификаторы томов МЛ для многотомных файлов. Необходимо, чтобы вначале первый том располагался на первом устройстве. Поиск на устройствах идет в порядке, определенном списком.

Пример:

```
MOU MT: A /VOL=(A,B,C)
```

/[NO]WAIT требует выдачи сообщения на консоль оператора при монтировании. Применяется при работе с удаленного терминала.

/[NO]WRITE разрешает или запрещает (/NOWRITE) операции записи на томе. Если запись разрешена, то проверяются привилегии доступа задачи.

Команда OPEN REGISTER (OPE п) позволяет вывести на терминал и, если необходимо, изменить содержимое слова из оперативной памяти. Из-за свопинга не разрешен доступ к памяти незафиксированных задач. Формат команды:

OPE[N] адрес [+ или —] смещ.[/кл.]

где адрес — восьмеричное число, определяющее адрес слова, считая

от начала памяти или от начала раздела при использовании любого ключа;

смещ.— смещение от точки, определяемой адресом. Ключи применяются только в системе с диспетчером памяти;

ключ /TASK=имя-задачи позволяет вывести слово из памяти раздела, занимаемого задачей; /PAR=имя раздела — из указанного раздела; /DRV=dd: — из памяти загружаемого драйвера; /KNL — из памяти супервизора.

После ввода командной строки OPE выводит строку следующего формата:

адрес/значение [СУВ] [значение-1] <завершающий-символ>

СУВ — символ управления выводом. Если СУВ не используется, то OPE выводит по умолчанию содержимое слова (значение) в восьмеричном виде. СУВ позволяет преобразовать это значение в той же строке в КОИ-7 (СУВ→"), в одиночные символы КОИ-7 (▼), в код RADIX-50(%) в десятичное число (.). В одной строке можно несколько раз пользоваться СУВ, но после каждого СУВ следует ввести <ESC>;

Завершающий символ: <ВК> — вывести следующее слово, ∧<ВК> — вывести предыдущее слово,*<ВК> или @<ВК> — содержимое данного слова рассматривать как адрес и распечатывать содержимое этого адреса, ><ВК> — содержимое слова рассматривать как инструкцию перехода и распечатать слово по адресу перехода. Здесь <ВК> — возврат каретки, <ESC> — единственный способ завершить OPEN. Кроме того, СУВ <ESC> изменяют формат вывода данного слова.

Для модификации слова следует после значения ввести новое значение.

Пример:

```
OPE 2000 <BK>
00002000/50313 37 <BK>
00002002/22237 ∧ <BK>
00002000/00037   <ESC>
```

Команда PARTITIONS (PAR нп) выводит на терминал детальную информацию обо всех разделах системы. Формат команды:

PAR[TITIONS]

Команда REA [*SSIGN*] (*REA* п) переназначает номер LUN указанной задачи на устройство ddnn:, которое может быть физическим, логическим или псевдоустройством. Формат команды:

REA [SSIGN] имя-задачи LUN ddnn:

Команда REDIRECT (*RED* п) позволяет переадресовать весь ввод-вывод с одного устройства на другое (подменить устройство). Нельзя переадресовать терминал, связанный с TI:, присоединенное устройство, устройство, содержащее смонтированный том. Формат команды:

RED [IRECT] новое-устр:=старое-устр:

Команда REMOVE (REM п) удаляет из памяти блок управления задачей TCB, созданный командой INSTALL. Файл образа не изменяется, и задачу можно вновь установить, если это необходимо. Формат команды:

REM[OVE] имя-задачи

Команда RESUME (RES нп/п) позволяет продолжить выполнение приостановленной задачи. Формат команды:

RES[UME] имя-задачи

Команда RUN используется для активизации задачи. В ОСРВ применяются 5 форматов команды RUN:

1. Немедленный запуск:

RUN имя-установленной-задачи[/UIC=[g, m]]

где UIC=[g, m] — привилегированный ключ, определяющий, под каким кодом UIC будет запущена задача. Этот код UIC будет определять привилегии доступа задачи.

2. Запустить через указанный в команде промежуток времени (привилегированная функция):

RUN имя-уст.-задачи m $\left\{\begin{matrix}T\\S\\M\\H\end{matrix}\right\}$

[/RSI=n $\left\{\begin{matrix}T\\S\\M\\N\end{matrix}\right\}$][/UIC=[g, m]]

где m величина промежутка времени, через который задачу следует запустить. Исчисляется от настоящего момента;

T (тики), S (сек), M (мин), H (часы) — единицы измерения. Промежуток времени не может быть больше 24 ч;

/RSI — ключ, позволяющий организовать циклическое выполнение задачи с интервалом

$$n=\left\{\begin{matrix}T\\S\\M\\H\end{matrix}\right\}$$

В последнем случае задача помещается в таймерную очередь, откуда ее можно удалить по команде CANCEL.

3. Запустить задачу через указанный промежуток времени после поступления события, заданного операндом СИНХ (п):

RUN имя-задачи СИНХ [промежуток] [/RSI=n {T S M H}][/UIC=[g, m]]

СИНХ определяет событие, которое наступает в конце часа (задается H), минуты (M), секунды (S) или тика (T). Остальные параметры вводятся так же, как в формате 2.

4. Запустить задачу в заданное время дня (п):

RUN имя-задачи время-дня [/RSI=n {T S M H}][/UIC=[g,m]]

Время дня задается в следующем виде hh:mm:ss

5. Активизация задачи в режиме установка-запуск-удаление:

RUN[⊙]спецификация-файла-образа-задачи /ключ(и)

Если ddnn: в спецификации файла не задано, то используется SY:. Если ⊙ задан, поиск образа ведется на LB: в системном каталоге. Ключи, применяемые в рассматриваемой команде, имеют тот же смысл, что и ключи команды INSTALL, так как эта команда является составной частью команды RUN пятого формата.

Ключи имеют следующие значения:

/CKP={YES NO};

/CMD="командная-строка"

передает строку символов запускаемой задаче;

/EST={YES NO}

определяет взаимоотношения породившей и порожденной задач: /EST=NO — нет синхронизации между задачами; /EST=YES — породившая задача ждет завершения порождаемой задачи, а при запуске — анализирует успешность этой операции;

/INC=n, /IOP={YES NO}, /PAD=имя, /PMD={YES NO}

/PRI=n,/SLV={YES NO}, /TASK=имя, /UIC=[g,m] —

эти ключи рассмотрены при описании других команд.

Команда SAVE (SAV п) записывает действующую в настоящее время версию ОСРВ в тот файл образа системы, из которого она была ранее загружена. Предыдущее содержимое файла теряется. Для успешного выполнения команды необходимо, чтобы SAVE вводилась с устройства CO:, задачи не имели незавершенных операций ввода-вывода, не было задач в файлах свопинга, не было смонтированных томов. Формат команды:

SAV[E] [/WB] [/ключ(и)]

где /WB требует установления связи данной версии файла образа с аппаратным загрузчиком;

/MOU=«строка» позволяет автоматически выдать команду MOU на системный диск при загрузке. Строка, заключенная в кавычки, должна соответствовать синтаксису команды MOU.

/SFILE=«спецификация-файла» позволяет задать командный файл, который будет выполнен при загрузке ОСРВ.

Команда SET (SET п/нп) предназначена для отображения и изменения параметров и характеристик устройств, задач, ОСРВ в процессе работы системы. Формат команды:

SET/ключ

В команде допустим только один ключ. Ключи имеют следующие значения:

/ABAUD [=TTnn:] (п) применяется только для полнодуплексного драйвера (ПД) терминала и позволяет автоматически определить скорость передачи по линии, через которую подключен указанный в команде терминал, и в соответствии с этой скоростью синхронизировать работу интерфейса. Если TTnn: не указан, распечатывается информация обо всех терминалах, работающих в этом режиме.

/NOABAUD[=TTnn:] (п) запрещает режим автоматического определения скорости передачи и применяется только для ПД.

/BRO [=TTnn:] (п) разрешает прием сообщений, выданных с помощью команды BRO, или распечатывает список терминалов, которым разрешен прием сообщений.

/NOBRO [=TTnn:] (п) запрещает прием сообщений, выданных с помощью BRO.

/BUF=ddnn: [размер] устанавливает (привилегированная функция) или распечатывает (непривилегированная функция) размер буфера по умолчанию для устройства ddnn:

CLI=TTnn: [имя-CLI] определяет (привилегированная функция) пользовательский интерпретатор CLI для указанного терминала. До команды SET следует выдать команду INS/CLI=YES. Если имя не указано, распечатывается имя CLI для данного терминала. Непривилегированный пользователь может определить CLI только для своего терминала.

/CRT [=TTnn:] устанавливает указанный терминал как дисплей или распечатывает список терминалов, являющихся дисплеями.

/NOCRT [=TTnn:] — указанный в команде терминал не является дисплеем.

/DCL [=TTnn:] разрешает прием команд DCL на заданном терминале или распечатывает список терминалов, принимающих команды DCL.

/EBC [=TTnn:] — драйвер терминала будет передавать задаче 8-битный код вместо обычного КОИ-7. Применяется только для ПД.

/NOEBC [=TTnn:] отменяет режим 8-битного кода. Здесь и далее предполагается, что если устройство не указано в команде, то выводится список устройств, на которые действует рассматриваемый ключ.

/ECHO [=TTnn:] требует эхо-отображения всех символов, вводимых с терминала TTnn:

/NOECHO [=TTnn:] запрещает эхо-отображение при запланированном вводе.

/ESCSEQ [=TTnn:] разрешает драйверу терминала принимать ESC-последовательность с TTnn:

/NOESCSEQ [=TTnn:] запрещает прием ESC-последовательности.

/FDX [=TTnn:] устанавливает полнодуплексный режим работы для заданного терминала. Используется только для ПД.

/NOFDX [=TTnn:] устанавливает полудуплексный режим.

/FORMFEED [=TTnn:] сообщает драйверу, что TTnn: имеет аппаратные средства для перевода формата.

/NOFORMFEED [=TTnn:] не имеет аппаратных средств для перевода формата.

/HFILL=TTnn: [m] определяет количество символов-наполнителей, которые помещаются за <ВК> при выводе, где $0 \leqslant m \leqslant 7$.

/HTT[=TTnn:] разрешает драйверу использование аппаратных средств горизонтальной табуляции. Используется только для ПД.

/NOHTT [=TTnn:] запрещает использование аппаратной табуляции. Используется только для ПД.

/HOLD [=TTnn:] разрешает постраничный вывод.

/NOHOLD [=TTnn:] запрещает постраничный вывод.

/LINES=TTnn: [m] определяет число строк на странице.

/LOGON (п) разрешает вход в систему.

/NOLOGON (п) запрещает вход пользователей в систему.

/LOWER [=ddnn:] запрещает преобразование русских букв в латинские на устройстве ddnn:

/NOLOWER [=ddnn:] автоматически преобразует русские буквы в латинские.

/MAIN=имя-раздела [:база:длина:тип] (п) устанавливает указанный раздел в качестве главного с заданной в команде базой (в 64-байтных блоках), длиной (в 64-байтных блоках) и типом (TASK — раздел, управляемый пользователем, SYS — системно-управляемый раздел, COM — общий раздел, DEV — раздел для доступа к странице внешних устройств).

/NOMAIN=имя раздела (п) удаляет указанный раздел из системы.

/MAXEXT [=n] (п) определяет, до какого размера задача может быть увеличена по директиве EXTEND TASK в системе с диспетчером памяти. n задается восьмеричным или десятичным числом, определяющим размер в 64-байтных блоках. Можно задавать размер в килобайтах (nK или n. K).

/MAXPKT [=n] (п) определяет максимальное количество элементов очереди ввода-вывода $0 \leqslant n \leqslant 15$.

/MCR [=TTnn:] разрешает распознавание команд MCR с данного терминала, применяется в системе с несколькими CLI.

/NETUIC=[g, m] устанавливает сетевой код UIC для ПП СЕТЬ СМ.

/POOL [=вершина] (п) увеличивает размеры динамической памяти. Вершина — это первый адрес, который доступен для разделов пользователей. Вершина задается числом, указывающим адрес в 64-байтных блоках.

/PRIV [=TTnn:] (п) присваивает TTnn: статус привилегированности.

/NOPRIV [=TTnn:] (п) отменяет статус привилегированности для TTnn:.

/PUB [=ddnn:] в системе с многопользовательской защитой устанавливает устройство ddnn: как общее.

/NOPUB [=ddnn:] отменяет статус устройства как общего.

/REMOTE [=TTnn:] [скорость] определяет, что TTnn: подключен к линии связи через адаптер или мультиплексор. „Скорость" устанавливает скорость в бодах.

/NOREMOTE [=TTnn:], где TTnn: локальный терминал.

/RPA [=TTnn:] разрешает драйверу «прозрачное чтение», при этом все управляющие символы записываются в буфер терминала. Применяется для ПД.

/NORPA [=TTnn:] запрещает операцию «прозрачное чтение». Применяется для ПД.

/SLAVE [=TTnn:] устанавливает для TTnn: статус подчиненности.

/NOSLAVE [=TTnn:] снимает статус подчиненности.

/SPEED [=TTnn:] [скорость-передачи:скорость-приема:] устанавливает скорости приема (передачи) для терминалов, подключенных через линии типа DZ или DH.

/SUB=имя-главного-раздела: имя-подраздела [:база-длина] устанавливает подраздел, заданный именем, базой и длиной внутри существующего главного раздела. База и длина задаются в 64-байтных блоках.

/NOSUB=имя-раздела: имя-подраздела удаляет подраздел из системы.

/SYSUIC [=[g, m]] определяет или распечатывает текущий системный код UIC. По умолчанию UIC=[1,54] для систем с диспетчером памяти и [1,50] для систем без диспетчера памяти.

/TERM=TTnn: [значение] задает тип терминала. Тип задается при генерации системы (например, VT05B или LA305). Используется для ПД. Возможно использование трех ключей вида: /ASR33=TTnn:/LA305=TTnn:/VT05B=TTnn: для полнодуплексного драйвера терминала. Эти ключи оказывают такое же действие, как и /TERM.

/TOP=имя-системно-управляемого-раздела: значение позволяет переместить верхнюю границу указанного раздела вверх или вниз. Размеры раздела изменяются, даже если в нем имеются установленные задачи.

Значение может быть задано в пяти форматах:

+n увеличивает раздел на n 64-байтных блоков;

−n уменьшает раздел;

+* или * перемещает границу раздела, увеличивая его, насколько возможно;

— * уменьшает размеры раздела насколько это возможно. Верхний адрес раздела перемещается к вершине первой фиксированной задачи или памяти, занимаемой драйвером. Если в разделе нет таких задач, его размер уменьшается до одного 64-байтного блока.

n задает размер раздела в 64-байтных блоках.

/TYPEAHEAD [=TTnn: [размер]] используется только для ПД. Требует сохранения введенных символов в буфере терминала перед передачей их в буфер пользователя. В результате символы, введенные между запросами, не теряются. Размер определяет длину буфера, по умолчанию — $(68)_{10}$.

/NOTYPEAHEAD [=TTnn:] отменяет промежуточную буферизацию.

/UIC [=[g, m] [:TTnn:]] устанавливает или отображает код UIC по умолчанию для данного терминала или указанного. Если команда введена с привилегированного терминала, она изменяет и регистрационный код UIC тоже, устанавливая привилегии доступа в соответствии с новым кодом UIC.

/VFILL [=TTnn:] требует от драйвера введения символов наполнителей. Используется только для ПД.

/NOVFILL [=TTnn:] запрещает драйверу использование символов-наполнителей. Используется только для ПД.

/WCHK [=ddnn:] требует проверки правильности записи любого блока на диске ddnn:. Проверка выполняется с помощью контрольного чтения. Ключ замедляет работу с устройством.

/NOWCHK [=ddnn:] запрещает проверку записи.

/WRAP [=TTnn:] инструктирует драйвер о необходимости добавления символов <CR> и <LF>, если число выведенных символов превышает размер буфера, установленный для TTnn:.

/NOWRAP [=TTnn:] отменяет выдачу <CR> <LF> при исчерпании буфера TTnn:

При работе с ключами, распределяющими оперативную память, необходимо учитывать следующие ограничения:

1. Система блокирует попытку удалить раздел или подраздел, если в нем имеется установленная задача, загружаемый драйвер и т. д.

2. Если удаляется главный раздел, автоматически удаляются все подразделы.

3. Блокируется удаление главного раздела, если установлена хотя бы одна задача в подразделе.

4. Имена разделов и подразделов должны быть уни-

кальными в ОСРВ. Запрещается наложение разделов или подразделов.

5. Память, выделенная под динамический пул, никогда не может быть использована для размещения разделов.

6. Нельзя выделять подразделы в системно-управляемом разделе, так как это делается самой системой в динамическом режиме.

Команда TAL (TAL нп) выводит на терминал информацию о всех задачах, установленных в системе (или об указанной) в формате команды ATL:

TAL [имя-задачи]

Команда TASKLIST (TAS нп) выводит детальную информацию о всех задачах (или об указанной), установленных в системе. Формат команды:

TAS[KLIST] [имя-задачи] [ddnn:]

Если ddnn: задано, то выводится информация только о тех задачах, которые установлены с ddnn:

Команда TIME (TIM п/нп) позволяет установить дату и время (П) или отобразить их. Формат команды:

TIM[E] [часы: мин [:сек]] [мес-1/день/год]

или

TIM[E] [часы:мин [:сек]] [день/мес-2/год]

где мес-I — месяц в виде числа $1 \leqslant$ мес $\leqslant 12$
мес-2 — месяц в виде трехсимвольной аббревиатуры.

Команда UFD (*UFD* нп/п) строит каталог пользователя на томе, отвечающем требованиям файловой системы ОСРВ, и вводит ссылку на новый каталог UFD в главный каталог MFD. UFD может быть создан в любой момент работы системы. Формат команды:

UFD ddnn: [метка-тома] [g ,m] [/ключ(и)]

где ddnn: — устройство, на котором создается UFD. Если метка опущена, проверка не производится. Непривилегированный пользователь может выдать UFD только на личное устройство;
[g, m] — код UIC, для которого строится UFD.

Ключи имеют следующие значения:

/ALLOC=n задает число заголовков в каталоге, выделяемое при создании UFD; по умолчанию — 32_{10}.

/PRO=[сист, влад, гр, общ.] определяет привилегии доступа к файлу каталога. Для каждого элемента можно задать полный или сокращенный набор привилегий (RWED); по умолчанию

/PRO=[RWED, RWED, RWED, R]

Команда UNBLOCK (UNB п/нп) позволяет продолжить выполнение ранее заблокированной (см. BLK) задачи. Формат команды:

UNB [LOCK] [имя-задачи]

Если имя не указано, команда использует имя TTпп:

Команда UNFIX (UNF п) освобождает задачу, ранее зафиксированную в памяти командой FIX. Формат команды:

UNF[IX] [имя-задачи]

Команда UNLOAD (UNLп) выгружает загружаемый драйвер, помещенный в память командой LOA. Формат команды:

UNL[OAD] dd:

Драйвер не будет выгружен, если на обслуживаемом им устройстве имеются незавершенные операции ввода-вывода, смонтированные тома или если устройство присоединено.

Команда UNSTOP (UNS нп/п) позволяет продолжить выполнение задачи, остановленной в результате выдачи макрокоманд RCST⊙ или STOP⊙. Формат команды:

UNS[TOP] [имя-задачи]

Если имя не указано, используется TTпп.

4.3. ИНТЕРПРЕТАТОР КОМАНДНЫХ СТРОК

Интерпретатор командных строк (CLI) является задачей, обслуживающей незапланированный ввод с терминала. При генерации системы можно указать число одновременно работающих интерпретаторов (CLI), которое не должно быть больше 15. По умолчанию программа MCR является интерпретатором CLI в ОСРВ. Перед использованием интерпретатора CLI его следует установить по команде MCR INS с ключом /CLI=YES.

Управление работой пользовательского интерпретатора осуществляется с помощью команды CLI программы MCR.

Команда CLI в ОСРВ имеет два формата:

CLI /ключ=имя-CLI
CLI/INI[T] =имя-CLI [/подключ(и)]

Мнемоника всех ключей может быть сокращена до первых трех символов.

Ключи имеют следующие значения:

/DISABLE =имя-CLI запрещает работу интерпретатора CLI, имя которого указано в ключе;

/ELIM =имя-CLI
/ELIM=*

удаляют структуры данных указанного интерпретатора CLI или всех CLI.

После выдачи CLI с ключом/ ELIM может быть выдана команда MCR REMOVE. Ключ/ELIM второго формата должен использоваться во время процедуры завершения работы системы.

/ENABLE =имя-CLI разрешает работу интерпретатора CLI, ранее остановленного по ключу/DISABLE.

/INIT =имя-CLI инициирует структуры данных указанного интерпретатора CLI. После этого терминалы могут быть присоединены к нему.

Подключи имеют следующие значения:

/CPR =«строка» указывает подсказку, которая будет выдана в ответ на CTRL/C. Необходимо предусмотреть выдачу перевода строки (<12>) и возврата каретки (<15>) в «строке». Если указана пустая строка (/CPR=""), подсказка не выводится.

/DISABLE инициирует интерпретатор CLI в режиме запрещения подключения терминалов. CLI начнет работу после выдачи команды CLI/ENABLE= имя-CLI.

/DPR=«строка» задает подсказку по умолчанию, которая выводится при вводе пустой командной строки. Пустая строка не дает возможности выдать подсказку.

/LGO указывает, что интерпретатор CLI должен принимать команды с терминала, на котором не зарегистрировался пользователь.

/MESSAGE — интерпретатор CLI должен получать сообщения о своем подключении или отключении, подключении или отключении терминалов к данному интерпретатору CLI.

/NULL — интерпретатор CLI будет принимать пустые командные строки, которые передаются при вводе возврата каретки в качестве первого знака в строке. Подсказка не выдается.

/PRIV — пользователь должен быть привилегированным для подключения к данному интерпретатору CLI.

/PROMPT — интерпретатор CLI требует подключения запроса на выдачу подсказки. По умолчанию CLI не извещается о подсказке.

/QUETET — регистрация и выход из системы без вывода сообщения на терминал.

/RESTRICT указывает, что только интерпретатор CLI может присоединить терминал. Ключ не может быть использован, если имя интерпретатора CLI имеет вид: ...xxx.

/MESSAGE =имя-CLI: «текст-сообщения» — указанное сообщение передается интерпретатору CLI.

/SHOW выводит информацию о каждом интерпретаторе CLI в системе.

/UNOVR аннулирует запрещение на подключение терминала к интерпретатору CLI. После входа в систему по HELLO гарантирует передачу команд только CLI.

4.4. ДИСПЕТЧЕР НЕУСТАНОВЛЕННЫХ ЗАДАЧ *TDX*

Диспетчер неустановленных задач (TDX) предоставляет пользователю удобные средства для запуска неустановленных по команде INS задач и для работы с файлами.

Для того чтобы использовать TDX, необходимо построить диспетчер неустановленных задач при генерации системы. После установки TDX имеет имя ...CA и принимает псевдокоманды, которые после необходимого преобразования передаются MCR.

Псевдокоманда DIR [спецификация-файла] преобразуется в команду:

```
PIP TI: = [спецификация-файла] /LI
```

Псевдокоманда TYP спецификация-файла преобразуется в команду:

```
PIP TI: = спецификация-файла
```

Псевдокоманда DEL [спецификация-файла] преобразуется в команду:

```
PIP спецификация-файла /DE
```

Псевдокоманда CHD преобразуется в команду SET/UIC, а CHD g, m — в команду

```
SET /UIC = [g ,m]
```

Псевдокоманда TDX печатает идентификацию задачи TDX. Если TDX не идентифицирует свою команду, то она передает MCR команду вида:

RUN⊙xxx/TASK=xxxTnn/CMD=«yy...y»

где xxx — имя задачи, yy...y — командная строка задачи.

4.5. КОСВЕННЫЕ КОМАНДНЫЕ ФАЙЛЫ

Косвенные командные файлы — это текстовые файлы, содержащие команды, адресованные конкретной задаче и интерпретируемые только ею. В ОСРВ различают *косвенные командные файлы задач* и *программы связи с оператором* MCR. Первые содержат информацию в терминах соответствующей задачи, вторые — в терминах процессора косвенных командных файлов AT.

Все косвенные командные файлы имеют тип CMD (например, ABC.CMD). Для активизации косвенного командного файла, направляемого AT, следует выдать командную строку вида:

```
@имя-файла.CMD/кл
```

или

```
@имя-файла/кл
```

Процессор косвенных командных файлов считывает файл и интерпретирует каждую строку или как команду, адресованную MCR или CLI пользователя, или как собственную директиву. Каждой директиве предшествует точка.

AT. допускает использование следующих ключей в командной строке.

/[NO]TR разрешает или запрещает (/NOTR) отладочную распечатку последовательно выполняемых директив AT. (по умолчанию — /NOTR).

/[NO]DE требует или запрещает удаление командного файла после выполнения. Если выполнение файла прервано по STOP или «/», то он в любом случае не стирается.

/[NO]MC передает или не передает команды, не опознанные AT., пользовательскому интерпретатору CLI (по умолчанию — /MC).

/[NO]CLI — синоним ключа/[NO]MC (по умолчанию CLI).

Набор директив AT. представляет собой процедурный язык косвенных командных файлов.

AT. оперирует с тремя типами переменных: логическими, числовыми и строковыми. *Логические* принимают значения «истина» или «ложь», *числовые* — целочисленные значения от 0 до 65535_{10}, *строковые* включают от 0 до 132_{10} байт в символьном коде.

Числовые переменные или константы могут объединяться с другими числовыми переменными или константами в *выражения* путем использования знаков арифметических операций: +, —, *, /. Логические выражения

строятся при помощи операций OR(!), AND(&), NOT(#). При этом не разрешается использование пробелов или знаков табуляции между операторами. Арифметические выражения вычисляются слева направо, если не используются скобки, которые изменяют порядок выполнения операций. Числовая переменная является десятичной или восьмеричной в зависимости от того, какое последнее значение ей было присвоено.

Имена всех *переменных* — это набор символов в коде КОИ-7, который начинается с заглавной латинской буквы (от A до Z) или знака ⊙. Остальные символы могут быть цифро-буквенными или содержать знак ⊙. Имя может содержать до шести символов.

Директивы языка косвенных командных файлов

Директивы отделяются от аргументов хотя бы одним пробелом или знаком горизонтальной табуляции. На каждой строке может быть записана только одна директива. Метки, применяемые для изменения порядка следования директив, располагаются в начале строки и заканчиваются двоеточием. Допустима только одна метка в строке.

Ниже приводится описание директив языка косвенных командных файлов.

Директива .ASK имеет форматы:

.ASK ssssss строка-символов
.ASK [значен.-по-умолчанию: тайм-аут] ssssss строка-символов
.ASK [:тайм-аут] ssssss строка-символов

Она выводит на терминал указанную в команде строку символов (вопрос) и в зависимости от ответа оператора присваивает логической переменной ssssss значение «истина» или «ложь». Если разрешено использование тайм-аута и, если он задан в директиве, AT. ждет ответа не более промежутка времени, заданного тайм-аутом, после чего присваивает переменной значение по умолчанию.

Тайм-аут задается в следующем виде: *nm*, где *n* — величина тайм-аута, а *m* — единица измерения этой величины — H (часы), M (минуты), S (секунды), T (тики).

Форматы ответов пользователя на вопрос директивы .ASK: Y<CR> присваивает ssssss значение «истина»; N<CR> присваивает значение «ложь», <CR> присваивает значение по умолчанию или, если оно не задано — «ложь»; <ESC> — «истина».

Пример:

.ASK III DO YOU WANT TO INSTALL PIP

AT. добавляет к вопросу символ «*» в начале строки и знак «?» — в конце вместе с вариантами ответа, т. е.

```
*DO YOU WANT TO INSTALL PIP? [Y/N]:
```

Директива .ASKN имеет форматы:

```
.ASKN ssssss строка-символов
.ASKN [ниж. гр: верх-гр: знач по умолч: тайм-аут] ssssss строка-символов
```

Директива ASKN позволяет присвоить переменной ssssss числовое значение. При использовании второй формы AT. проверит также, лежит ли введенное значение в интервале верхняя граница — нижняя граница.

Пример:

```
.ASKN[3:36:26:30S] AA DEFINE NUM AA
```

AT. выведет строку

```
*DEFINE NUM AA[O R:3—36 D:26 T:30S]:
```

где O определяет тип переменной AA — восьмеричный по умолчанию;
R:3—36 — допустимый интервал значений AA;
D:26 — значение по умолчанию,
T:30S — тайм-аут.

Директива .ASKS имеет форматы:

```
.ASKS ssss строка-символов
.ASKS[min-дл:max-дл:строка-по-умолч:тайм-аут] ssssss строка-символов
```

.ASKS позволяет ввести с терминала строку символов и присвоить строковой переменной ssssss соответствующее значение.

Пример:

```
.ASKS T ENTER TIME
*ENTER TIME [S]:
```

где s — строковая переменная.

Директива .BEGIN обозначает начало блока вида .BEGIN — .END

Метки и локальные переменные, определенные внутри блока, являются неопределенными вне блока.

Директива .CHAIN имеет формат:

```
.CHAIN имя-косвенного-ком.-файла/кл
```

позволяет продолжать обработку, используя новый командный файл.

Ключи, которые могут быть заданы в директиве .CHAIN, аналогичны ключам, описанным при активизации командного файла.

Директива .CLOSE имеет формат:

```
.CLOSE[#n]
```

Эта директива закрывает вторичный файл, открытый директивой .OPEN. Необязательный параметр #n указывает номер закрываемого файла. $0 < n < к - 1$, где к — количество открытых файлов.

Директива .DATA имеет формат:

```
.DATA [#n] строка-символов
```

.DATA позволяет записать одну строку символов, длиной до 132_{10} в предварительно открытый по директиве .OPEN вторичный файл. Если открыто несколько файлов, то #n определяет, в какой (по порядку открытия) из них следует записать строку символов.

Пример:

```
.SETS ABC "BEGIN"
.OPEN KK
.DATA 'ABC'
.CLOSE.
```

В этом примере в файл KK.DAT записывается одна строка, содержащая слово BEGIN.

Директива .DEC имеет формат:

```
.DEC ssssss
```

Она вычитает 1 из числовой переменной ssssss. AT. аварийно завершится, если ssssss описана как строковая или логическая переменная.

Директива .DELAY имеет формат:

```
.DELAY n {H}
         {M}
         {S}
         {T}
```

Директива приостанавливает выполнение командного файла на указанное параметром n число часов, минут, секунд или тиков.

Директивы .ENABLE, .DISABLE разрешают или запрещают использование заданных в директиве параметров и имеют форматы:

```
.ENABLE параметр [,параметр] ...
.ENABLE DATA [#n]
.DISABLE параметр [,параметр] ...
```

Параметры:

ATTACH — присоединение терминала при отображении комментариев, по умолчанию действует .ENABLE ATTACH;

CONTROL/Z — использование CTRL/Z в качестве ответа на вопросы по директивам типа .ASKx. Если этот

параметр запрещен, CTRL/Z рассматривается как конец файла, и AT. заканчивает работу. По умолчанию действует DISABLE CONTROL/Z.

DATA — вывод данных во вторичный файл. Строки, следующие за .ENABLE DATA, записываются в ранее открытый вторичный файл (#n определяет в какой именно) до появления директивы .DISABLE DATA.

GLOBAL — использование глобальных символов. .ENABLE GLOBAL — позволяет рассматривать все символы, начинающиеся со знака ⊙, как глобальные, определенные на всех уровнях вложенности командных файлов. По умолчанию действует .ENABLE GLOBAL.

DECIMAL — интерпретация всех цифровых символов по умолчанию как десятичных. По умолчанию действует .DISABLE DECIMAL, т. е. интерпретация всех цифровых символов, как восьмеричных.

DELETE — стирание командного файла после выполнения. По умолчанию стирание запрещено, действует DISABLE DELETE.

MCR — передать все командные строки, неопознанные AT., CLI пользователя или MCR. По умолчанию действует .ENABLE MCR.

LOWERCASE — преобразование русских букв в латинские. По умолчанию действует .ENABLE LOWERCASE.

SUBSTITUTION — замена. Если замена разрешена, AT. заменяет строковые переменные, заключенные в апострофы, на их значения. По умолчанию замена запрещена, действует .DISABLE SUBSTITUTION.

ESCAPE — использование <ESC> в качестве ответа на .ASK.<ESC> присваивает соответствующей переменной значение «истина». По умолчанию .DISABLE ESCAPE запрещает .AT принимать <ESC> как ответ на .ASK.

ESCAPE-SEQ — разрешение AT. принимать ESC-последовательность с терминала. По умолчанию прием ESC-последовательности запрещен, действует .DISABLE ESCAPE-SEQ.

OVERFLOW — переполнение при арифметических операциях в числовых выражениях. По умолчанию переполнение запрещено — действует .DISABLE OVERFLOW.

QUIET — не выводить эхо-сопровождение команд CLI, выданных из командного файла. По умолчанию вывод производится — действует .DISABLE QUIET.

TIMEOUT — использование тайм-аутов в директивах

типа .ASKx. По умолчанию использование тайм-аутов разрешено .ENABLE TIMEOUT.

TRACE — трассировка. По умолчанию действует .DISABLE TRACE.

TRUNCATE — усекание строк. Если усекание разрешено, AT. игнорирует ошибки, связанные с превышением допустимой длины строки — 132 символа. По умолчанию усекание запрещено, действует DISABLE TRUNCATE.

Пример:

```
.ENABLE SUBSTITUTION
.ASKS NAM ENTER FILE NAME
MAC 'NAM' = 'NAM'
```

При выполнении командного файла:

```
> *ENTER FILE NAME [S]: AABB ⟨ВК⟩
> MAC AABB=AABB
```

Директива .ERASE уничтожает все локальные и глобальные определения символов или указанное определение глобального символа. Форматы директивы:

```
.ERASE LOCAL
.ERASE GLOBAL
.ERASE SYMBOL глобальный-символ
```

Если .ERASE LOCAL задана внутри блока .BEGIN — .END, то она уничтожает только локальные символы данного блока. Автоматически уничтожаются определения локальных символов при использовании файла другого уровня вложенности.

Директива .EXIT. завершает обработку в командном файле данного уровня и передает управление в файл более высокого уровня.

```
.EXIT [значение]
```

Значение необязательного параметра присваивается специальной переменной ⟨EXSTAT⟩.

Директива .GOSUB имеет формат:

```
.GOSUB метка [параметры]
```

Директива вызывает запоминание состояния обработки командного файла (для последующего возврата) и выполняет переход на метку, заданную в .GOSUB. Эта метка может располагаться даже внутри блока .BEGIN—.END. Метка в директиве .GOSUB указывается без ведущей точки и не сопровождается двоеточием (например, GOSUB MET1).

Параметры, если они имеются, становятся значением специальной переменной — ⟨COMMAN⟩. Для разбора параметров следует использовать директиву .PARSE.

Программа, на которую передается управление по директиве .GOSUB, должна заканчиваться директивой .RETURN.

Директива .GOTO — безусловный переход на метку AT. ищет метку для выполнения перехода только внутри того блока .BEGIN—.END, в котором стоит .GOTO. Таким образом, нельзя передать управление в другой блок или выйти за пределы данного. Формат директивы:

```
.GOTO метка
```

Пример:

```
.GOTO 100
```

Директива .IF имеет формат:

```
.IF логическое-выражение директива-1
```

.IF инициирует проверку значения логического выражения, заданного в директиве. Если значение выражения — «истина», то выполняется директива-1. В противном случае анализируется следующая строка.

Логическое выражение может включать стандартные операции сравнения: EQ, NE, GE, LE, GT, LT. При сравнении строковых переменных анализируются попарно символы слева направо. При первом несовпадении операция заканчивается.

Директива .IFACT/.IFNACT позволяет проверить, активна задача или бездействует. При соблюдении условия выполняется директива, находящаяся в той же строке. Формат директивы:

```
.IFACT имя-задачи директива-1
.IFNACT имя-задачи директива-1
```

Пример:

```
IFACT PIP .GOTO 300
```

Директива .IFDF/.IFNDF проверяет, определена ли логическая, строковая или числовая переменная. Форматы директивы:

```
.IFDF ssssss директива-1
.IFNDF ssssss директива-1
```

Директива .IFINS/.IFNINS проверяет, установлена ли указанная в директиве задача. Если проверка успешна, выполняется директива-1. Форматы директивы:

```
IFINS имя-задачи директива-1
IFNINS имя-задачи директива-1
```

Пример:

.IFNINS PIP INS ⊙PIP

Директива .IFENABLE/.IFDISABLE имеет форматы:

.IFENABLE параметр директива-1
.IFDISABLE параметр директива-1

Параметры рассмотрены при описании директивы .ENABLE.

Директива .IFLOA/.IFNLOA проверяет, загружен ли указанный загружаемый драйвер. Если проверка успешна, выполняется директива-1. Форматы директивы:

.IFLOA dd: директива-1
.IFNLOA dd: директива-1

Пример:

.IFNLOA LP: LOA LP:

Директива .IFT/IFF проверяет значение указанной логической переменной и имеет форматы:

.IFT ssssss директива-1
.IFF ssssss директива-1

Директива-1, указанная в строке, выполняется, если ssssss имеет значение «истина» (1-я строка) или «ложь» (2-я строка).

Замечание. Несколько директив типа .IF могут быть использованы для проведения сложной проверки путем объединения их с помощью логических функций .OR (или) и .AND (и). Другие обозначения ! и &.

Пример:

.IFT A .OR(.IFT B.AND.IFF C) .GOTO 5

Директива .INC увеличивает на единицу числовую переменную ssssss. Формат директивы:

.INC ssssss

Директива .ONERR позволяет передать управление на подпрограмму обработки ошибок, выявленных при работе АТ. Директива применяется только для обработки следующих ошибок:

задача не установлена (.XQT, .WAIT);
переменная не определена;
синтаксические ошибки (.XQT, .WAIT, .DELAY);
ошибочная команда;
ошибка замены;
ошибка в типе переменной (.IF, .IFT, .IFF, .INC, .DEC);
некорректная попытка изменения типа переменной (.ASKx, .SETx);

ошибка при работе с файлом.
Формат директивы:

ONERR метка 1

Подпрограмма, начинающаяся меткой, работает со специальной переменной <ERRCTL>. Такая подпрограмма должна завершаться выдачей сообщения и директивой .EXIT.

Директива .OPEN открывает вторичный файл для записи. Формат директивы:

.OPEN [#n] спецификация-файла

где n — номер открываемого файла.

Тип файла по умолчанию — DAT, номер — 0.

Директива .OPENA открывает вторичный файл для дополнения. Данные добавляются в конце существующего файла. Формат директивы:

.OPENA [#n] спецификация-файла

где n — номер открываемого файла.

Директива .OPENR открывает вторичный файл для чтения

OPENR[#n] спецификация-файла

где n — номер открываемого файла.

Директива .PARSE позволяет произвести разбивку командной строки на требуемое количество подстрок. Формат директивы:

PARSE строковая-переменная строка-управления переменная 1, ..., переменная n

Строковой переменной должно быть присвоено значение командной строки, которую следует разбить на подстроки. Строка управления — это набор знаков-разделителей, заключенных в кавычки. Разделители (чаще всего — запятые или пробелы) отделяют друг от друга параметры в командной строке или команду от параметров. Если разделителей указано меньше, чем имеется параметров в строке, последний разделитель используется многократно. Переменная-1, ..., переменная-n — это строковые переменные, которые получают значения подстрок в результате разбивки. Переменная получает значение «пусто», если соответствующая подстрока пуста. Специальная переменная <STRLEN> после разбивки содержит действительное число подстрок в строковой переменной.

Пример:

```
.PARSE <COMMAN> "␣", A B C D E
```

Пусть <COMMAN> содержит AAA 1,37,, DFC; тогда в результате разбивки переменная A содержит AAA, B — 1, C — 37, D — «пусто», E — DFC.

Директива .PAUSE прекращает выполнение косвенного командного файла до момента вмешательства оператора. После вмешательства интерпретация возобновляется с точки приостановки. Формат директивы:

```
.PAUSE
```

Директива .READ имеет формат:

```
.READ[#] ssssss
```

Она считывает очередную запись из открытого файла, заданного параметром #n (см. директиву OPEN) и присваивает строковой переменной ssssss содержимое этой записи. Символ <EOF> принимает значение «истина», если обнаружен конец файла, <FILER> содержит код ошибки файловой системы ОСРВ. Ошибка может быть зафиксирована в процессе операции «Чтение».

Директива .RETURN реализует возврат из подпрограммы и имеет формат:

```
RETURN
```

Директива .SETT/.SETF/.SETL имеет форматы

```
.SETT ssssss
.SETF ssssss
.SETL ssssss !!!!!!
```

Указанные директивы определяют или изменяют значение логической переменной ssssss. Если до директивы SETx переменная не была определена, то директива определит ее и присвоит значение «истина» (.SETT) «ложь» (.SETF) или то значение, которое определит логическое выражение *!!!!!!*, AT. аварийно завершится, если ssssss была ранее определена как числовая или строковая переменная.

Пример:

```
.SETL A B!ORCC
```

Директива .SETN определяет или изменяет значение числовой переменной и имеет формат:

```
SETN ssssss числовое-выражение
```

Пример:

```
SETN ABC 3*(B7—5)
```

Директивы .SETO/.SETD переопределяют или определяют впервые числовую переменную как восьмеричную (.SETO) или десятичную (.SETD) с необходимым пересчетом значения. Формат директивы:

```
.SETO ssssss
.SETD ssssss
```

Пример:

```
SETN A 10 — присваивает A значение 10₈
SETD A — объявляет A десятичной переменной со знаком (8₁₀)
SETO A — объявляет A восьмеричной переменной со знаком (10₈)
```

AT. аварийно завершится, если ssssss ранее была определена как строковая или логическая переменная.

Директива .SETS переопределяет значение строковой переменной или определяют ее вновь и присваивает ей указанное значение. Формат директивы:

```
SETS ssssss строковое-выражение
```

Строковое выражение может включать конкатенацию строковых переменных и строковых констант. Переменные могут входить в выражение полностью или частично. Строковые константы заключаются в кавычки ("). Знак конкатенации — плюс (+). При необходимости включить в выражение часть строковой переменной применяется обозначение вида: ssssss [m: n], где m — номер символа строковой переменной, начиная с которого следует извлекать символы для формирования константы; n — номер последнего символа. Вместо n можно указать *, что означает «включить до конца».

Пример:

```
SETS A1 "ABCDEF"
SETS A2 A1[3:4]
.SETS A3 A1[1:3] + "KM" + A2
```

A3 имеет значение ABCKMCD

Директивы .STOP и/. .STOP немедленно прекращает обработку командного файла. AT. выводит сообщение @ EOF. Формат директивы:

```
STOP [числовое-выражение]
```

Числовое выражение передается AT. в качестве кода завершения. /вызывает те же действия, что и .STOP без параметра.

Директива TEST. Применяется два формата этой директивы. Первый:

```
.TEST ssssss
```

анализирует ssssss и в результате присваивает символам значения:

1) если ssssss — строковая переменная, то ⟨SYMTYP⟩ присваивается значение 4, ⟨STRLEN⟩ содержит длину строки. Кроме того, переменные ⟨ALPHAN⟩, ⟨NUMBER⟩, ⟨RAD50⟩, ⟨OCTAL⟩ получают значения в зависимости от характеристик ssssss;

2) если ssssss — числовая переменная, ⟨SYMTYP⟩ присваивается значение 2;

3) если ssssss — восьмеричная переменная, ⟨SYMTYP⟩ присваивается значение 2, а ⟨OCTAL⟩ — «истина»;

4) если ssssss — логическая переменная, ⟨SYMTYP⟩ получает значение 0.

Второй формат:

```
.TEST строка подстрока
```

где строка — строковая переменная или константа, подстрока — строковое выражение.

AT. ищет подстроку внутри строки. Если находит, то ⟨STRLEN⟩ указывает на номер знака строки, с которого начинается подстрока. В противном случае ⟨STRLEN⟩ обнуляется.

Директива TESTDEVICE позволяет получить информацию о любом устройстве системы. Соответствующая информация становится значением строковой переменной ⟨EXSTRI⟩. Элементы этой строки разделены запятыми, что облегчает анализ с помощью директив .PARSE и .TEST. Формат директивы:

```
TESTDEVICE dd[nn]:
```

где dd[nn]: — физическое имя устройства.

В результате выполнения директивы ⟨EXSTRI⟩ имеет формат:

```
ddnn:, xx1, xx2, xx3, xx4, атрибут 1, ..., атрибут n
```

где xx1 ... xx4 — содержимое полей U.CW1...U.CW4 из UCB устройства:
атрибуты: NSD — такого устройства нет в системе, LOD — драйвер загружен, UNL — драйвер не загружен, ONL — устройство в рабочем состоянии, OFL — устройство в автономном режиме, MTD — устройство смонтировано, NMT — устройство не смонтировано или это немонтируемое устройство, FOR — том смонтирован, как созданный на другой системе, NFO — устройство немонтируемое или том не смонтирован, как созданный на другой системе, PUB — общее устройство, NPO —

не общее устройство. ATT — присоединено к другой задаче, ATU — присоединено к данному AT., NAT — не присоединено, ALO — зарезервировано за другим терминалом, ALU — зарезервировано данным терминалом, NAL — свободно.

Директива .TESTFILE имеет два формата. Первый формат:

```
TESTFILE llnn:
```

где llnn: логическое имя устройства. AT. находит связанное с ним физическое устройство и помещает его имя в ⟨FILSPC⟩, а код ошибки системы управления файлами в ⟨FILERR⟩.

Второй формат:

```
TESTFILE спецификация-файла
```

Директива формирует в ⟨FILSPC⟩ полную спецификацию файла, указанного в .TESTFILE.

Пример:

```
TESTFILE AA: INN.TXT
 FILERR  =1 успешное завершение
⟨FILSPC⟩=DK1:[100, 100]INN.TXT; 17
```

где AA: — логическое имя устройства, назначенное на DK1:

Директива .TESTPARTITION позволяет получить информацию об указанном разделе. Формат директивы:

```
TESTPARTITION имя-раздела
```

Строковая переменная ⟨EXSTRI⟩ содержит в результате следующую информацию: имя раздела, база, длина, тип. База и длина измеряются в 64-байтных блоках. Если искомого раздела нет в системе, то вместо типа передается NSP.

Директива .WAIT приостанавливает работу AT. до завершения указанной задачи. Формат директивы:

```
WAIT имя-задачи
```

Если имя задачи не указано, используется TTnn:

⟨EXSTAT⟩ содержит код завершения задачи.

Директива .XQT позволяет запустить несколько задач, не дожидаясь завершения ранее запущенных. Обычно AT. запускает задачу и ждет ее завершения. Формат директивы:

.XQT имя-задачи командная-строка-задачи

Пример:

```
.XQT MAC A=A
.XQT PIP A.TSK; */DE
```

5

СОЗДАНИЕ ЗАДАЧ

5.1. ФУНКЦИИ ПОСТРОИТЕЛЯ ЗАДАЧ

Единицей работы в ОСРВ является *задача.* Она состоит из одного или нескольких объектных модулей, скомпонованных в *загрузочный модуль — образ задачи.* Процесс создания задачи осуществляется построителем задач ТКВ, который обеспечивает три основные функции: компоновку (связь) объектных модулей, назначение адресов для образа задачи и создание структур данных для задачи.

При *компоновке объектных модулей* ТКВ создает единый файл образа задачи, определяет значения всех глобальных символов и ссылок на библиотеки объектных модулей.

В системе без диспетчера памяти задача связывается с базовым адресом раздела, для которого она строится, и не может быть установлена и запущена на выполнение в разделе с иным базовым адресом. В системе с диспетчером памяти (ДП) задача строится с виртуального нулевого адреса и, благодаря аппаратуре диспетчера памяти, может перемещаться в памяти.

Структуры данных, создаваемые ТКВ, при построении задачи используются при установке и выполнении задачи.

5.2. СТРУКТУРА ЗАДАЧИ

Файл образа задачи, создаваемый ТКВ, содержит (в простейшей форме) метку, занимающую группу блоков, заголовок задачи, образ памяти задачи. Для выгружае-

мых задач файл образа задачи может содержать область выгрузки.

Метка содержит данные, используемые при установке задачи в системе: имя задачи и параметры раздела, в котором задача должна выполняться, размер задачи и ее приоритет, таблицу назначений логических устройств.

При установке задачи эта информация используется для создания блока управления задачей и инициализации заголовка задачи.

Заголовок задачи содержит информацию, которую управляющая программа использует при выполнении задачи. Заголовок загружается в память вместе с образом задачи и сохраняется в файле выгрузки с образом задачи при выгрузке задачи.

ТКВ создает и частично инициализирует заголовок задачи, а при установке задачи инициализируется остальная часть заголовка.

Память задачи включает скомпонованные объектные модули программ (т. е. коды инструкций и данные), а также *стек задачи.* Стек задачи является областью памяти задачи, которая может использоваться для временного хранения информации и связи подпрограмм. Обращение к стеку осуществляется через указатель стека SP.

Процесс трансляции исходной программы и связи объектных модулей построителем задач определяется контекстом программных секций.

Программная секция является единицей памяти задачи, содержащей коды инструкций и данные, и имеет следующие характеристики:

имя программной секции;

набор атрибутов, определяющих тип секций, режим доступа, размещение и возможность перемещения в памяти;

длину, определяющую размер памяти, резервируемой для программной секции.

Программные секции создаются с помощью директивы макроассемблера .PSECT или автоматически с помощью специальных операторов на языках высокого уровня (например, COMMON в программах на Фортране).

Атрибуты программных секций и их значения приведены в табл. 5.1.

Таблица 5.1

Атрибуты программных секций

Атрибут	Обозначение	Значение
1	2	3
Код доступа к секции	RW	Чтение и запись. Данные можно считывать из программной секции и записывать в нее
	RO	Только чтение. Данные можно считывать из программной секции, но записывать в нее нельзя
Код типа секции	D	Данные. Программная секция содержит данные
	I	Инструкции. Программная секция содержит инструкции и данные или только инструкции
Код области действия	GBL	Глобальный. Имя программной секции распознается за пределами сегмента. Построитель задач распределяет память для такой программной секции, исходя из ссылок за пределами определяющего сегмента перекрытия
	LCL	Локальный. Имя программной секции распознается только внутри определяющего сегмента перекрытия. Построитель задач распределяет память для программной секции, исходя из ссылок только внутри определяющего сегмента
Код сохранения	SAV	Сохранение. Программная секция с атрибутом SAV размещается внутри корневого сегмента
Код размещения секции	CON	Конкатенация. Программные секции с одинаковыми именами дадут конкатенацию. Общая необходимая память есть сумма размеров всех программных секций с одним именем
	OVR	Перекрытие. Программные секции с одинаковыми именами перекрывают друг друга. Размер общей памяти равен размеру самой большой по длине программной секции с этим именем
Код перемещаемости	REL	Перемещаемый. Базовый адрес программной секции вычисляется относительно виртуального базового адреса задачи
	ABS	Абсолютный. Базовый адрес программной секции не является перемещаемым и всегда равен нулю

5.2.1. ПРИВИЛЕГИРОВАННЫЕ ЗАДАЧИ

Привилегированность задачи определяется ее правами доступа, кроме памяти своего раздела, к памяти управляющей программы и внешней странице, а также возможностью выполнения операций ввода-вывода логических блоков на монтированных томах. Непривилегированная задача имеет доступ только к памяти своего раздела и, возможно, к разделяемым областям. В сис-

темах без диспетчера памяти любая задача имеет доступ ко всей физической памяти.

Построение привилегированной задачи осуществляется с помощью ключа /PR:n (см. п. 5.6), где аргумент n может принимать значения 0, 4, 5. Значение аргумента n определяет номер первой пары регистров ДП, используемых в качестве базовых для отображения образа задачи.

При указании ключа/PR:0 задача строится как привилегированная, но не имеет доступа к внешней странице и к памяти управляющей программы. Задача отображается с нулевого виртуального адреса и может иметь максимальный размер 32 Кслова.

Привилегированная задача, построенная с ключом PR:0, имеет возможности использования директивы изменения приоритета (ALTP⊙), указания имени устройства в директивах порождения задач, выполнения операций ввода-вывода логических блоков на монтированных томах. Такая задача выполняется только в режиме пользователя и не может переключаться в системный режим.

Привилегированные задачи, построенные с ключами /PR:4 или /PR:5, кроме отмеченных возможностей для задач, построенных с ключом /PR : 0, имеют доступ к внешней странице памяти и к памяти управляющей программы. Задача, построенная с ключом /PR:4, отображается с виртуального адреса 100000_8 и может иметь максимальный размер 12 Кслов. Задача, построенная с ключом /PR:5, может иметь максимальный размер 8 Кслов и отображается с виртуального адреса 120000_8. Внешняя страница при использовании привилегированной задачи отображается с виртуального адреса 160000_8. Привилегированные задачи, построенные с ключами /PR:4 или /PR:5, могут переключаться из режима пользователя в системный режим.

5.3. ПОСТРОЕНИЕ ЗАДАЧ С ПЕРЕКРЫТИЯМИ

Структура задач с перекрытиями обеспечивает уменьшение размеров физической памяти или виртуального адресного пространства, требуемого задачами при их выполнении. Для создания задачи с перекрытиями необходимо разделить задачи на *сегменты*: единственный *корневой сегмент*, который находится постоянно в памяти, и необходимое число *сегментов перекрытий.*

Каждый сегмент является единицей загрузки и состоит из набора программных секций. Сегменты, которые

перекрывают друг друга, должны быть логически независимыми, т. е. не содержать взаимных ссылок. Возможны два типа сегментов перекрытий (в дальнейшем просто перекрытий): перекрытия, резидентные на диске, и перекрытия, резидентные в памяти.

Разделяемым ресурсом для перекрытий, резидентных на диске, являются физическая память и виртуальное адресное пространство. Задачи, использующие этот тип перекрытий, называются *задачами с диск-резидентными перекрытиями.*

Перекрытия, которые в процессе выполнения задачи постоянно находятся в физической памяти, называются *перекрытиями, резидентными в памяти.* Они загружаются с диска только при первом обращении к ним, а затем остаются в памяти на все время выполнения задачи. Для перекрытий, резидентных в памяти, разделяемым ресурсом является только виртуальное адресное пространство. Задачи с такой структурой перекрытий выполняются быстрее, чем задачи с перекрытиями, резидентными на диске, так как в этом случае не требуется загрузка перекрытий с диска.

Использование перекрытий, резидентных в памяти, возможно только в системах с диспетчером памяти и при поддержке управляющей программой директив управления памятью.

5.3.1. СТРУКТУРА ПЕРЕКРЫТИЙ, РЕЗИДЕНТНЫХ НА ДИСКЕ

Рассмотрим структуру задачи TSK с перекрытиями, резидентными на диске. Задача TSK строится из четырех вводных файлов. Каждый вводный файл состоит из единственного объектного модуля с тем же именем, что и файл.

Задача строится по команде:

```
>TKB TSK=CNTRL, A, B, C
```

Модуль CNTRL для этой задачи определяется корневым сегментом, модули A, B, C определяются логически независимыми сегментами перекрытий.

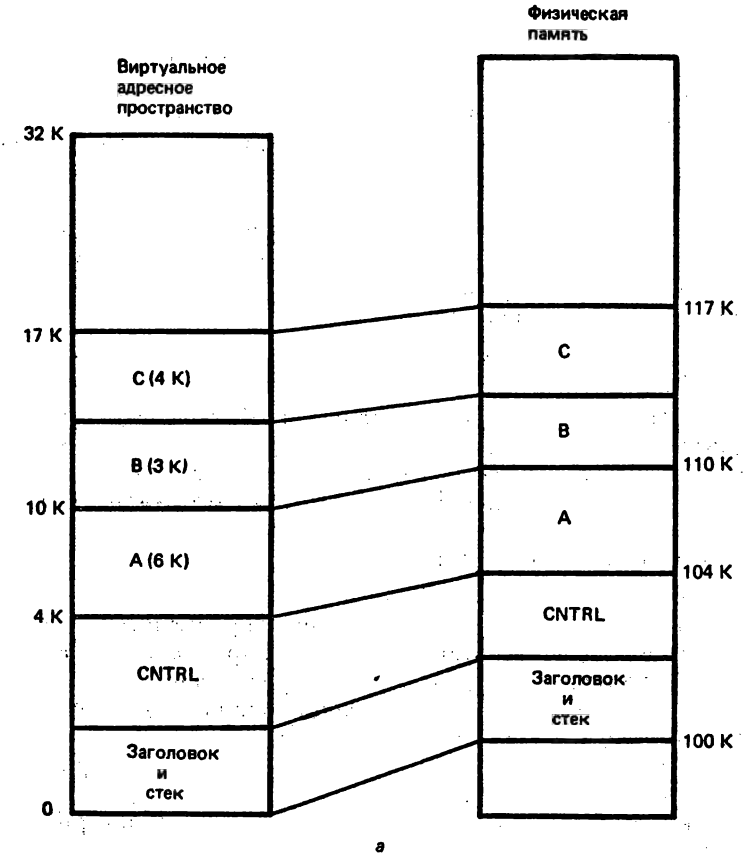
Перекрытия A, B, C для такой задачи будут разделять между собой физическую память и виртуальное адресное пространство. Введение структуры перекрытий, резидентных на диске, уменьшает для задачи TSK общую потребность в памяти, как показано на рис. 3 (с. 139—140).

Как видно из рис. 3, размер оперативной памяти, требуемой для задачи TSK как односегментной, опреде-

ляется суммой длин всех модулей задачи — CNTRL, A, B, C.

Для задачи TSK, построенной как многосегментная задача с перекрытиями, резидентными на диске, требуемый размер оперативной памяти определяется длиной корневого сегмента CNTRL и длиной наибольшего сегмента перекрытий A. Кроме памяти, требуемой для размещения модулей, для многосегментной задачи необходима память для организации управления структурой перекрытий.

Разделяя сегменты перекрытий A, B, C на логически независимые части, возможно еще больше сократить требуемый объем памяти для задачи TSK.



а

Рис. 3. Распределение памяти для односегментной (*а*) и многосегментной (*б*) задачи TSK (начало)

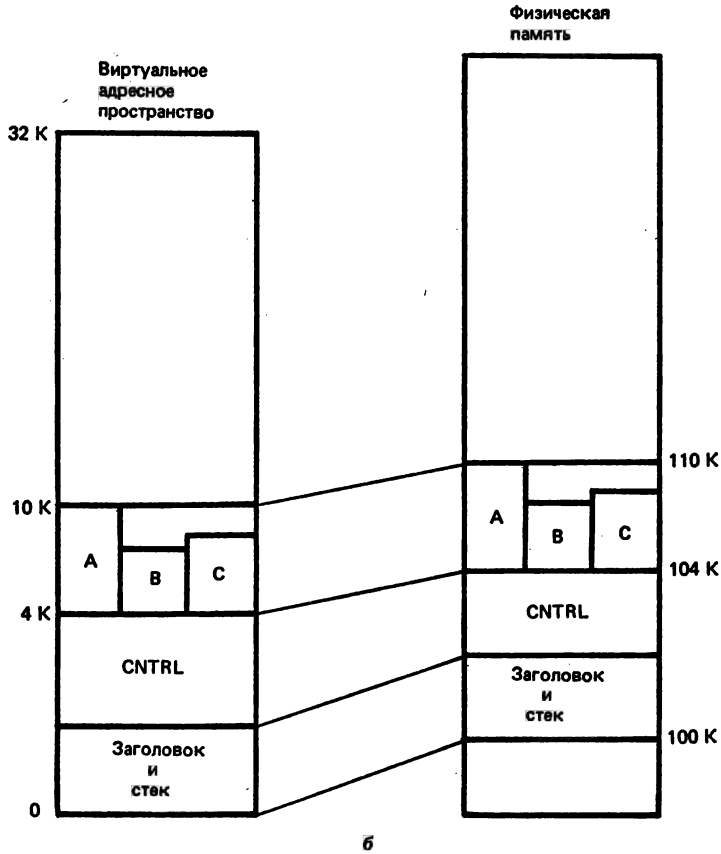


Рис. 3. Распределение памяти для односегментной (*а*) и многосегментной (*б*) задачи TSK (окончание)

5.3.2. СТРУКТУРА ПЕРЕКРЫТИЙ, РЕЗИДЕНТНЫХ В ПАМЯТИ

Перекрытия, резидентные в памяти, загружаются с диска только при первом обращении к ним, а затем постоянно находятся в памяти, разделяя между собой виртуальное адресное пространство. В отличие от перекрытий, резидентных на диске, перекрытия, резидентные в памяти, не разделяют физической памяти, так как они располагаются в отдельных областях памяти, выровненных на границу 32-словного блока. Процесс загрузки перекрытий, резидентных в памяти, связан поэтому (в отличие от собственно загрузки перекрытий, резидентных

на диске) с отображением разделяемых виртуальных адресов в непрерывные области физической памяти, содержащие перекрытия. На рис. 4 представлено отображение сегментов перекрытий, резидентных в памяти.

Длина каждого сегмента перекрытий, резидентного в памяти, должна выбираться по возможности кратной 4 Ксловам. Это определяется тем, что распределение виртуального адресного пространства, связанного с аппаратурой диспетчера памяти, осуществляется страницами по 4 Кслова.

В целях уменьшения требуемой физической памяти задачи целесообразно размещать общие библиотечные объектные модули в сегменте, доступном для всех ссылающихся сегментов.

Резидентные библиотеки в целях экономии виртуальной памяти целесообразно разбивать на сегменты, резидентные в памяти.

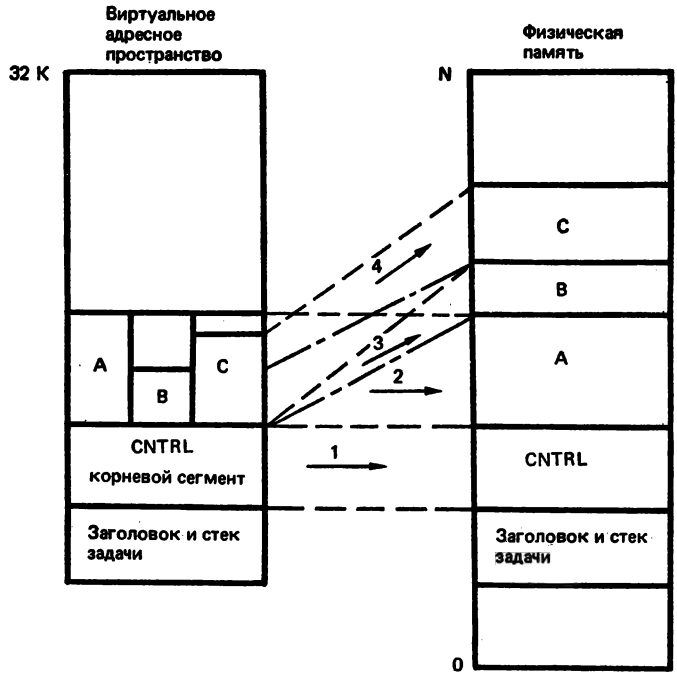


Рис. 4. Отображение сегментов перекрытий, резидентных в памяти, в различные моменты времени (1, 2, 3, 4)

5.3.3. ДЕРЕВО ПЕРЕКРЫТИЙ

Структура задачи с перекрытиями может быть представлена в виде дерева перекрытий. Каждая *ветвь дерева* представляет собой сегмент. Параллельные ветви одного уровня соответствуют сегментам, которые перекрывают друг друга и должны быть логически независимыми. Примыкающие ветви соответствуют сегментам, которые не разделяют виртуальное адресное пространство или физическую память. Для этих сегментов не требуется логической независимости.

Построитель задач включает *язык описания перекрытий* для представления структуры задач с перекрытиями в виде одного или нескольких деревьев. Структура, содержащая несколько деревьев, является *сложным деревом*. Сложные деревья перекрытий состоят из *главного дерева* и одного или нескольких *поддеревьев*. Корневой сегмент главного дерева загружается управляющей программой при активизации задачи, а все остальные сегменты загружаются в соответствии с вызовами.

Особенностью структуры сложного дерева является возможность доступа из одного поддерева ко всем модулям в главном дереве и других поддеревьях. Это объясняется тем, что каждое дерево размещается в собственной памяти.

На рис. 5 изображена структура задачи с перекрытиями, описываемая в виде единственного дерева перекрытий. Дерево имеет корневой сегмент CNTRL, три главных ветви A0, B0, C и шесть листьев — A1, A21, A22, B1, B2, C.

Дерево имеет столько *путей*, сколько у него *листьев*. Путь вниз определяется как путь от листа к корню (например, A22—A2—A0—CNTRL); путь вверх определяется как путь от корня к листу (например, CNTRL—A0—A1). Понятия дерева и пути являются важными для понимания механизма загрузки и разрешения глобальных символов.

Модули могут вызывать другие модули, которые находятся на том же самом пути. На рис. 5 модуль CNTRL является общим для всех путей дерева и поэтому может вызывать любой модуль в дереве и быть вызванным из любого модуля.

Если модуль одного сегмента перекрытий вызывает модуль из другого сегмента, этот сегмент должен уже находиться в памяти и быть отображенным или должен загружаться в память.

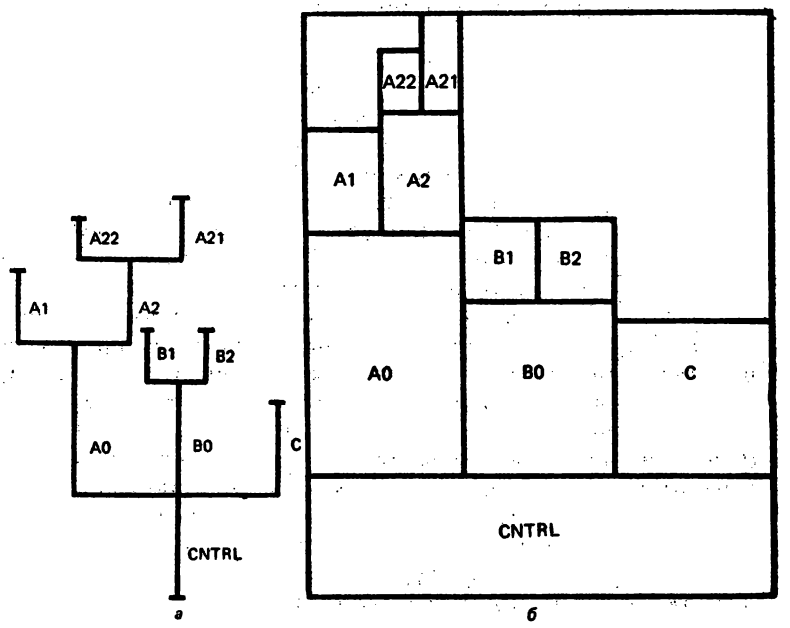


Рис. 5. Структура задачи с перекрытиями:
а — дерево перекрытий для задачи;
б — диаграмма распределения памяти

Процедура для разрешения глобальных символов в многосегментной задаче может быть описана следующим образом:

1) построитель задач выбирает сегмент перекрытий для обработки в соответствии с размещением сегментов относительно корневого сегмента: обработка начинается с сегментов, наиболее удаленных от корня;

2) каждый модуль в сегменте просматривается для поиска глобальных определений и ссылок;

3) для каждого глобального определения символа построитель задач отыскивает ссылки на этот символ во всех сегментах на путях, проходящих через обрабатываемый сегмент; аналогичным способом для глобальных ссылок осуществляется поиск соответствующих определений;

4) если глобальный символ определен несколько раз на разных путях и к нему имеется ссылка из сегмента, расположенного на общей части этих путей, символ считается неопределенным;

5) если глобальный символ определен несколько раз на одном пути, символ считается многократно определенным;

6) два или более определений глобального символа, находящихся на разных путях и не имеющих ссылок к ним из сегмента, общего для этих путей, являются *допустимыми.*

5.3.4. ЯЗЫК ОПИСАНИЯ ПЕРЕКРЫТИЙ

Язык описания перекрытий ODL построителя задач позволяет описывать структуру перекрытий задачи. Описание перекрытий является текстовым файлом, содержащим директивы языка описания перекрытий. Этот файл имеет по умолчанию расширение ODL и указывается в командной строке построителя задач как единственный вводный файл с ключом/MP.

Язык описания перекрытий имеет пять директив для описания дерева перекрытий: .ROOT, .END, .PSECT, .FCTR, .NAME. Описание дерева может содержать только одну директиву .ROOT и одну директиву .END; директива .END должна быть последней в описании дерева.

Директива .ROOT определяет структуру задачи с перекрытиями и обычно является первой в описании перекрытий. Аргументами директивы .ROOT могут быть спецификации файлов, имена ветвей, имена сегментов, имена программных секций. Аргументы директивы .ROOT могут разделяться следующими операторами:

оператор «—» (тире) указывает конкатенацию, т. е. что перекрытия должны находиться в физической памяти одновременно;

оператор «,» (запятая) внутри скобок означает, что перекрытия разделяют физическую память или виртуальное адресное пространство;

оператор «,» (запятая) вне скобок разделяет деревья при описании структуры сложного дерева;

Оператор «!» (восклицательный знак) перед левой скобкой указывает перекрытия, заключенные в скобки, как резидентные в памяти;

круглые скобки используются для указания группы сегментов, которые начинаются с одного и того же виртуального адреса.

Директива .END должна быть последней и указывает построителю задач на завершение ввода описания перекрытия.

Описание перекрытий для задачи, структура которой представлена на рис. 5, может быть дано в следующем виде:

```
.ROOT CNTRL — (A0 — (A1, A2 — (A22, A21)), B0 — (B1, B2), C)
.END
```

Директива .FCTR используется для описания перекрытий в более наглядном виде, что осуществляется за счет возможности разбиения строки директивы .ROOT на несколько строк.

Например, описание перекрытий, приведенное выше, с использованием директивы .FCTR может быть сделано следующим образом:

```
        .ROOT CNTRL-(AFCTR, BFCTR, C)
AFCTR:  .FCTR A0-(A1, A2-(A21, A22))
BFCTR:  .FCTR B0-(B1, B2)
        END
```

Директива .FCTR может быть вложенной, максимальная глубина вложения — 16.

Директива .NAME является средством для объявления имени сегмента и присвоения сегменту необходимых атрибутов. Эта директива используется, чтобы обеспечить применимость метода автозагрузки для сегментов, не содержащих выполняемые инструкции, или для задания в явном виде имени сегмента, загружаемого методом ручной загрузки. Если директива .NAME не указывается, для идентификации сегмента используется имя первого файла или имя первой программной секции в сегменте. Формат директивы:

```
.NAME имя сегмента (атрибуты)
```

Атрибуты разделяются запятой и не являются обязательными. Значения атрибутов по умолчанию — NOGBL, DSK.

Атрибуты могут иметь следующие значения:

GBL — имя сегмента вводится в таблицу глобальных символов сегмента, что обеспечивает загрузку сегментов, не содержащих инструкций, методом автозагрузки;

NOGBL — имя сегмента не вводится в таблицу глобальных символов сегмента;

DSK — для заданного сегмента резервируется память на диске;

NODSK — для заданного сегмента не резервируется память на диске, что обеспечивает возможность отказа в выделении памяти в файле образа задачи для сегмента, не содержащего данных. Предполагается, что такой

сегмент будет заполняться данными только при выполнении задачи.

Директива .PSECT является средством для перемещения указанной явно глобальной программной секции в требуемый сегмент. Имя программной секции должно быть указано первым в параметрах директивы .PSECT, а остальные параметры являются атрибутами программной секции, перечисленными в табл. 5.1, и могут задаваться в произвольном порядке.

Изменим пример, приведенный при описании директивы .FCTR. Разместим в корневом сегменте еще одну программную секцию DATA с данными. Описание перекрытий будет иметь вид:

```
        .PSECT DATA, RW, D, GBL, REL, OVR

        .ROOT CNTRL - DATA - (AFCTR, BFCTR, C)
AFCTR:  .FCTR A0 - (A1, A2 - (A21 , A22))
BFCTR:  .FCTR B0 - (B1, B2)
        .END
```

Такое размещение программной секции DATA обеспечивает информационное взаимодействие всех сегментов перекрытий через общую область данных в корневом сегменте, который эту секцию не содержал.

5.3.5. МЕХАНИЗМЫ ЗАГРУЗКИ ПЕРЕКРЫТИЙ

Существуют два способа загрузки перекрытий в память — автозагрузка и ручная загрузка.

Автозагрузка перекрытий выполняется системой управления перекрытиями автоматически при передаче управления на автозагружаемый сегмент вверх по дереву. Автозагружаемые компоненты в описании перекрытий указываются с помощью индикатора автозагрузки — *.

Индикатор автозагрузки может быть указан для следующих компонентов: имени файла; части описания дерева перекрытий, заключенной в скобки; имени программной секции, содержащей инструкции; имени сегмента; имени ветви, указанной по директиве .FCTR. Индикатор автозагрузки отмечает первый компонент ветви как автозагружаемый. Если описание ветви заключено в скобки, все компоненты ветви являются автозагружаемыми.

При автозагрузке используется механизм загрузки по путям, т. е. вызов из сегмента другого сегмента вверх по дереву обеспечивает автоматически загрузку в па-

мять всех сегментов на пути от вызывающего до вызванного сегмента.

Указание индикатора автозагрузки (в рассмотренном выше примере для директивы .FCTR) перед скобками, содержащими имена ветвей, обеспечит автозагрузку всех компонентов в описании дерева перекрытий.

```
.ROOT CNTRL - *(AFCTR, BFCTR, C)
```

Автозагрузка сегментов осуществляется за счет создания построителем задач вектора автозагрузки для глобального символа, на который сделана ссылка.

Передача управления на глобальный символ заменяется вызовом подпрограммы автозагрузки ⊙AUTO. Путем размещения индикаторов автозагрузки только в точках, где загрузка действительно требуется, можно минимизировать число векторов автозагрузки, создаваемых для задачи. Для программных секций с атрибутом D (данные) вектор автозагрузки не создается.

Изменение механизма автозагрузки для сегмента, не содержащего выполняемых инструкций, осуществляется за счет объявления имени сегмента директивой .NAME с атрибутом GBL. При вызове такого сегмента с помощью директивы CALL он автоматически загружается, и управление возвращается в вызывающий сегмент. Ссылки на глобальный символ вверх по дереву разрешаются непосредственно в программной секции с атрибутом D.

Ручная загрузка сегментов обеспечивается включением в программу явных вызовов подпрограммы загрузки перекрытий ⊙LOAD. При ручной загрузке загружается только сегмент, указанный в запросе, т. е. загрузка по путям не осуществляется. Ручная загрузка может быть запрошена в синхронном или асинхронном режимах.

При программировании на макроассемблере ручная загрузка осуществляется вызовом подпрограммы ⊙LOAD. Формат макровызова:

```
MOV #PBLK, R0
CALL ⊙LOAD
```

где PBLK — адрес блока параметров, который содержит: длину блока параметров (от 3-х до 5 слов); номер флага события, задаваемый для асинхронной загрузки; имя загружаемого сегмента в формате RADIX-50; адрес двухсловного блока состояния ввода-вывода; адрес подпрограммы асинхронного прерывания по завершении загрузки сегмента.

Ручная загрузка в программах на Фортране осуществляется через подпрограмму MNLOAD, вызывающую под-

программу ⊙LOAD. Параметры, задаваемые в вызове подпрограммы MNLOAD, аналогичны параметрам в макровызове подпрограммы ⊙LOAD.

5.4. РАЗДЕЛЯЕМЫЕ ОБЛАСТИ

5.4.1. ОБЩИЕ СВЕДЕНИЯ

Разделяемая область представляет собой область оперативной памяти, содержащую коды инструкций или данные, которые могут совместно использоваться любым количеством задач. Разделяемые области обеспечивают:

уменьшение требуемого объема оперативной памяти за счет совместного использования одной копии подпрограммы несколькими задачами;

разделение общих данных между несколькими задачами, осуществляя таким образом информационную связь между задачами.

Непривилегированная задача может быть связана не более чем с семью разделяемыми областями.

Разделяемая область строится без заголовка и стека, так как она содержит невыполняемый образ. При построении разделяемой области должен быть обязательно создан, кроме файла образа, *файл определения символов*. Этот файл в последующем используется построителем задач для установления связи между разделяемой областью и задачей, использующей разделяемую область. С целью экономии виртуального адресного пространства (ВАП) задачи, связываемой с разделяемой областью, можно построить разделяемую область со структурой перекрытий, резидентных в памяти. Необходимо отметить, что распределение виртуального адресного пространства (ВАП) для разделяемых областей осуществляется с кратностью 4 Кслова, так как управление памятью связано с использованием аппаратуры диспетчера памяти (ДП).

Разделяемые области могут быть позиционно-независимыми или абсолютными. *Позиционно-независимые области* в системах с ДП могут размещаться в произвольных адресах виртуального адресного пространства (ВАП) задачи. *Абсолютные разделяемые области* связываются с фиксированными адресами виртуального адресного пространства задачи.

Разделяемая область может быть объявлена позиционно-независимой, если она содержит:

инструкции, которые будут правильно выполняться независимо от их размещения в виртуальном адресном пространстве ссылающейся задачи;

данные, которые не являются адресно-зависимыми.

При построении позиционно-независимых областей необходимо учитывать следующие особенности:

инструкции или данные разделяемой области не должны включаться в неименованную программную секцию (.BLK);

задача, использующая разделяемую область, не должна иметь программные секции с именами, совпадающими с именами программных секций разделяемой области;

порядок размещения программных секций (/SQ или /SG) для разделяемой области и ссылающейся к ней задачи должен быть одинаковым.

В файл определения символов для позиционно-независимой разделяемой области включаются определения всех программных секций области. За счет этого задача, связанная с такой разделяемой областью, может ссылаться к ней по имени программной секции.

Для абсолютной разделяемой области в файл определения символов включается только абсолютная секция .ABS. Ссылки к разделяемым инструкциям и данным в этой секции осуществляются по глобальному имени.

5.4.2. СОЗДАНИЕ РАЗДЕЛЯЕМОЙ ОБЛАСТИ

Для создания разделяемой области необходимо построить файл образа задачи и файл определения символов, используемый для связи ссылающейся задачи с разделяемой областью.

Абсолютная разделяемая область строится при указании ключа /—PI, а ключ /PI объявляет разделяемую область позиционно-независимой. При создании разделяемой области должен указываться всегда ключ /—HD.

В табл. 5.2 приведены различные варианты использования ключей /PI, /CO, /LI и их действие на файл распределения памяти.

Таблица 5.2

Варианты использования ключей и их действие на файл распределения памяти

Ключи, указанные совместно с /—HD	Разделяемая область		Имя программной секции	Размещение программных секций в файле определения символов	Размещение символов в файле распределения памяти
	абсолютная	позиционно-независимая			
/PI/LI		Да	Совпадает с именем корневого сегмента библиотеки	Одна перемещаемая секция	Все символы со значениями относительно начала секции
/PI/CO		Да	Включаются имена всех программных секций	Все программные секции перемещаемые	Все символы и программные секции
/—PI/LI или /LI	Да		·ABS	Одна абсолютная секция	Все символы абсолютные
/—PI/CO или /CO	Да		Включаются имена всех программных секций	Все программные секции абсолютные	Все символы абсолютные
/PI		Да	Соответствует /PI/CO		
/—PI	Да		Соответствует /—PI/LI		

Ниже приводится пример создания абсолютной резидентной библиотеки LIBR из файлов A, B, C:

```
>TKB
>LB:[1,1] LIBR/—HD/MM,LP:,LB:[1,1]LIBR=A,B,C
TKB/
ENTER OPTIONS:
TKB>STACK=0
TKB>PAR=LIBR:160000:1000
```

Так как область LIBR является абсолютной, то во всех ссылающихся на нее задачах она будет отображаться с виртуального адреса 160000.

После создания разделяемой области задачи могут связываться с ней. До установки и запуска задачи разделяемая область должна быть размещена в памяти с помощью команды оператора INS.

5.4.3. РАЗДЕЛЯЕМЫЕ ОБЛАСТИ С ПЕРЕКРЫТИЯМИ, РЕЗИДЕНТНЫМИ В ПАМЯТИ

Структура разделяемой области с перекрытиями, резидентными в памяти, определяется через файл описания перекрытий. При построении разделяемой области ТКВ не включает в файл образа области базовые данные перекрытий (описатели сегментов, векторы автозагрузки) и программы управления перекрытиями. Эти данные включаются в файл определения символов разделяемой области, который используется для связи со ссылающейся задачей.

При построении задачи в ее корневой сегмент включаются базовые данные и глобальные ссылки к программам управления перекрытиями.

Для подавления запросов на загрузку перекрытий каждый сегмент перекрытия разделяемой области создается с атрибутом NODSK.

С помощью необязательного параметра GBLREF можно включить глобальные ссылки в корневой сегмент разделяемой области. В этом случае необходимые векторы автозагрузки и определения генерируются без включения таких ссылок в объектный модуль.

Процедуру создания разделяемой области с резидентными перекрытиями можно представить следующим образом:

определить структуру перекрытий, содержащую только резидентные в памяти перекрытия;

включить параметр GBLREF или задать в корневом сегменте модуль, содержащий глобальные ссылки для определения точек входа тех сегментов перекрытий, для которых будут создаваться векторы автозагрузки и глобальные определения.

Ниже приводится пример создания разделяемой области с резидентными в памяти перекрытиями:

```
.ROOT A - !(*B,C - *D)
.NAME A
.END
```

Корневой сегмент A является пустым и имеет нулевую длину. Все выполняемые инструкции находятся в сегментах, резидентных в памяти, которые создаются из объектных файлов B, C, D, содержащих одноименные глобальные точки входа.

При построении разделяемой области в качестве

входного файла для TKB указывается файл, содержащий описание структуры перекрытий разделяемой области:

```
TKB> A/—HD/MM, LP:, SY:A=A/MP
ENTER OPTIONS:
TKB> GBLREF=B, C, D
TKB> PAR=A:160000:20000
TKB> STACK=0
TKB> //
```

При построении разделяемой области имя раздела, имя файла-образа области и имя файла определения символов должны быть одинаковыми.

В приведенном примере глобальные ссылки на точки входа B, C и D, включаемые в корневой сегмент, генерируют в файле определения символов для точки C соответствующее определение, а для точек B и D — векторы автозагрузки.

Файл определения символов также содержит базовые данные, необходимые системе управления перекрытиями, для загрузки перекрытий. Эти базовые данные передаются через файл определения символов, ссылающейся к разделяемой области задаче. Поэтому загрузка резидентных в памяти перекрытий для разделяемой области не отличается от загрузки перекрытий, резидентных в памяти, создаваемых непосредственно как образ задачи.

Необходимо учитывать следующие ограничения для разделяемых областей с резидентными в памяти перекрытиями:

разделяемая область не может использовать метод автозагрузки для ссылок на другие резидентные в памяти перекрытия, если имена сегментов не совпадают, другие перекрытия могут загружаться методом ручной загрузки;

задача не может ссылаться к именованной программной секции, находящейся в сегменте перекрытия разделяемой области; для возможности таких ссылок программная секция должна быть включена в корневой сегмент разделяемой области;

в отличие от перекрытий, резидентных в памяти задачи, число векторов загрузки внутри каждой ссылающейся задачи к разделяемой области не зависит от количества действительных ссылок на точки входа и может быть больше количества ссылок.

5.4.4. ГРУППОВЫЕ БИБЛИОТЕКИ

Термин «групповые библиотеки» обозначает одновременно функцию TKB и структуру, которую он создает. Групповые библиотеки позволяют динамически отобра-

жать разделяемые области, резидентные в памяти. Например, задача может использовать три библиотеки LIB1, LIB2, LIB3 по 16 Кбайт, отображая их через общее адресное окно. Таким образом, разделяемые библиотеки при выполнении задачи используют не 48 Кбайт, а разделят 16 Кбайт виртуального адресного пространства задачи.

Функция групповых библиотек реализуется ТКВ в два этапа. Первый этап обеспечивает установление связей между библиотеками (в случае, если в одной библиотеке имеются ссылки к другой библиотеке группы). Этот этап не зависит от механизма переотображения. Второй этап включает директивы управления памятью, необходимые для отображения соответствующих групповых библиотек.

Задача связывается с групповыми библиотеками, с перекрытиями, резидентными в памяти, с использованием необязательного параметра CLSTR.

5.4.5. ОБЩИЕ ПРАВИЛА ПОСТРОЕНИЯ ГРУППОВЫХ БИБЛИОТЕК

При построении разделяемых групповых библиотек необходимо выполнять следующие правила.

1. Все библиотеки, кроме первой, должны иметь резидентные перекрытия. Под первой библиотекой понимается первая именованная библиотека, заданная в необязательном параметре CLSTR. Первая библиотека может быть односегментной или с перекрытиями. Все библиотеки с перекрытиями, включая первую, должны иметь пустой корневой сегмент.

2. Ссылки между групповыми библиотеками разрешаются косвенно через механизм векторных вызовов. Векторный механизм реализуется в ссылающейся задаче и обеспечивает автозагрузку требуемого перекрытия с передачей управления на точку входа.

3. Переопределенные точки входов не должны появляться в файле определения символов последующих библиотек. Это может быть достигнуто исключением каждого переопределенного символа точки входа с помощью необязательного параметра GBLXCL.

4. Для вызываемых библиотечных процедур не должна использоваться передача параметров через стек. Это правило относится ко всем групповым библиотекам, кроме первой, заданной в необязательном параметре CLSTR.

5. Все групповые библиотеки должны быть позиционно-независимыми или построены с одинаковым базовым виртуальным адресом.

6. Передача управления по асинхронным и синхронным системным прерываниям в групповые библиотечные программы недопустима. Это связано с тем, что в момент прерывания требуемая библиотека может оказаться неотображенной.

5.5. КОМАНДНАЯ СТРОКА ТКВ

Командная строка для построителя задач в общем виде содержит спецификации выводных файлов, отделенных знаком равенства от спецификаций вводных файлов. В командной строке ТКВ можно указать до трех выводных файлов и любое число вводных файлов.

Список выводных файлов может содержать спецификации трех файлов в следующем порядке: файл образа задачи, файл распределения памяти, файл определения символов. Любой выводной файл может быть опущен, и тогда вместо него в командной строке должна быть указана запятая.

Файл образа задачи содержит образ задачи, который загружается в память и запускается на выполнение.

Файл распределения памяти является символьным, содержащим информацию о распределении памяти задачи, разрешении глобальных символов и при необходимости список перекрестных глобальных ссылок.

Файл определения символов является двоичным файлом, содержащим определения глобальных символов в задаче и их виртуальные или перемещаемые адреса. Файл определения символов используется как входной файл при создании разделяемых резидентных областей.

Командная строка ТКВ может быть представлена различными способами:

1) выводные файлы = вводные файлы;
2) = вводные файлы;
3) @ косвенный файл,

где каждый файл задается его спецификацией.

Командная строка ТКВ может задаваться в виде одной строки или в многостроковом формате при использовании большого числа вводных файлов. Необязательные параметры используются для указания характеристик строящейся задачи.

Признаком для завершения ввода командной строки

является знак «//», если нужно завершить работу ТКВ, или знак — «/», если предполагается использовать ТКВ для построения другой задачи.

Знак «/», введенный в начале строки за спецификацией вводных файлов, является указанием ТКВ ввести необязательные параметры.

Спецификация вводных и выводных файлов соответствует стандартным системным соглашениям ОСРВ и содержит: устройство, код идентификации, определяющий каталог, имя файла, тип файла, номер версии и любое число ключей. В общем виде спецификация файла имеет формат:

устр: [гр., чл] имя файла. тип; версия/ключ

Компоненты спецификации — устройство, код идентификации, тип, версия и ключи — являются необязательными. Значения этих компонентов по умолчанию приведены ниже, в табл. 5.3.

Пример командной строки для построения задачи с именем PRIM:

```
>TKB
TKB> PRIM, PRIM=IN1, IN2
TKB> IN3, IN4
> TKB> /
ENTER OPTIONS:
TKB>  STACK=1000
TKB>  PRI=100!COMMON=JRNL : RO
TKB> //
```

Таблица 5.3

Компоненты спецификации файла

Компоненты спецификации файла	Значения по умолчанию
устр:	Имя физического устройства, на котором смонтирован том, содержащий файл. По умолчанию используется устройство, указанное в предыдущей спецификации, если устройство ни в одной предыдущей спецификации не указано, выбирается SY0:
[гр., чл.]	Код идентификации UIC, задаваемый номером группы и номером члена в группе, используется для нахождения каталога пользователя UFD. По умолчанию используется код идентификации, указанный в предыдущей спецификации, или код UIC терминала, если UIC не был указан ни в одной предыдущей спецификации
Тип	TSK — для файла образа задачи; STB — для файла определения символов; MAP — для файла распределения

Продолжение

Компоненты спецификации файла	Значения по умолчанию
	памяти; OLB — для библиотеки объектных модулей; ODL — для описания перекрытий; CMD — для косвенного командного файла
Версия	Для вводного файла — версия с наибольшим номером из всех существующих файлов с указанным именем и типом, для выводного файла — версия на единицу больше максимальной

В приведенном примере ТКВ создает файл образа задачи PRIM.TSK и файл распределения памяти PRIM.MAP из вводных объектных файлов IN1.OBJ, IN2.OBJ, IN3.OBJ, IN4.OBJ. Для указания характеристик строящейся задачи используются необязательные параметры.

5.6. КЛЮЧИ ПОСТРОИТЕЛЯ ЗАДАЧ

В спецификации файлов командной строки ТКВ дополнительно можно задавать один или несколько ключей. Если ключи не задаются, то ТКВ использует их значения по умолчанию.

Ключ задается с помощью символьного кода (2—4 символа) с предшествующим знаком «/». Если ключ начинается со знака минус «—» или с символов NO, действие ключа, определенного следующими двумя символами, отвергается. Например, ключ CP может иметь одно из трех значений: /CP; /—CP; /NOCP. Каждый ключ может использоваться только с определенными типами вводных или выводных файлов.

В табл. 5.4 приведены ключи построителя задач, их описание и значение по умолчанию, а также типы файлов, с которыми эти ключи могут быть заданы: T — файл образа задачи; M — файл распределения памяти; S — файл определения символов; I — вводной файл.

Таблица 5.4

Ключи построителя задач

Формат	Значение	Тип файла	Значение по умолчанию
/AC [:n]	Задача является вспомогательным управляющим процессором. Такая задача является привилегированной; n указывает первую пару регистров диспетчера памяти для отображения задачи и может принимать значения 0, 4 и 5	T	/—AC n=5
/AL	Задача строится как выгружаемая в область выгрузки, выделенную в файле образа задачи	T	/—AL

Продолжение

Формат	Значение	Тип файла	Значение по умолчанию
/CC	Вводной файл является конкатенацией объектных модулей, которые все включаются в образ задачи. При отрицании ключа ТКВ включает в образ задачи только первый модуль файла	I	/CC
/CM	Резидентные в памяти сегменты перекрытий выравниваются на границу 256-словного блока для совместимости с возможностями директив управления памятью	T	/—CM
/CP	Задача строится как выгружаемая в системный файл выгрузки. Ключи /CP и /AL — взаимно исключающие	T	/—CP
/CR	Список глобальных перекрестных ссылок добавляется к файлу распределения памяти	M	/—CR
/CO	Строится разделяемая общая область, описание всех программных секций включается в файл распределения памяти. Используется вместе с ключом /—HD	T, S	/CO
/DA	Включить в задачу средства отладки. Если ключ указан с файлом образа задачи, включается стандартный отладчик LB0: [1,1] ODT.OLB, если ключ указан с вводным файлом, то вводный файл является отладочным средством	T, I	/—DA
/DL	Указанный с ключом вводный файл является библиотекой объектных модулей и заменяет системную библиотеку объектных модулей LB0:[1,1] SYSLIB.OLB для разрешения неопределенных глобальных ссылок	I	/—DL
/EA	Задача использует расширитель арифметики	T	/—EA
/FP	Задача использует процессор с плавающей запятой	T	/—FP
/FU	Разрешение неопределенных глобальных ссылок с помощью библиотеки объектных модулей, выбираемой по умолчанию, осуществляется для сегментов всех поддеревьев перекрытий. При отрицательном значении ключа разрешаются только ссылки из главного корневого сегмента и текущего дерева	T	/—FU
/HD	Заголовок задачи включается в образ задачи. При создании разделяемой области необходимо задавать /—HD	T, S	/HD
/IP	Отрицательное значение ключа /—IP указывает, что для привилегированной задачи не требуется отображение во внешнюю страницу памяти	T	/IP
/LB	Вводный файл является библиотечным. Если ключ используется без аргументов, то файл просматривается для разрешения неопределенных глобальных символов; ключ с аргументами указывает имена модулей, задаваемых как аргументы, которые должны быть включены в образ задачи	I	/—LB

Продолжение

Формат	Значение	Тип файла	Значение по умолчанию
/LI	Образ задачи является разделяемой библиотекой, ключ /LI должен указываться вместе с ключом /—HD	T, S	/—LI
/MA	В файл распределения памяти включается информация о файле. По умолчанию значением ключа для вводного файла является /MA, для файла распределения памяти /—MA	I M	/MA /—MA
/MM	Целевая система, в которой будут выполняться задачи, имеет диспетчер памяти (ДП). По умолчанию значение ключа устанавливается в зависимости от того, в какой системе строится задача — с ДП или без него	T	/MM /—MM
/MP	Вводный файл является файлом описания структуры перекрытий для строящейся задачи. Этот файл должен быть единственным вводным файлом в командной строке для ТКВ	I	/—MP
/NM	Запрещает печать диагностических сообщений о числе неопределенных символов в сегментах задачи и многократном определении программных секций	T	/—NM
/PI	Задача или разделяемая область содержит только позиционно-независимые коды	T, S	/—PI
/PM	При аварийном завершении задачи автоматически распечатывается содержимое памяти задачи	T	/—PM
/PR[:n]	Задача строится как привилегированная. Значение n может быть 0, 4, 5 и определяет первую пару регистров диспетчера памяти, используемых для отображения задачи	T	/—PR
/RO	Разрешается распознавание в файле описания перекрытий оператора резидентных в памяти перекрытий—«!». При отрицательном значении ключа /—RO проверяется корректность использования этого оператора, но перекрытия строятся как диск-резидентные	T	/RO
/SE	Сообщения могут посылаться задаче из другой задачи с помощью системной директивы SEND⊙	T	/SE
/SG	Программные секции группируются в соответствии с кодами доступа (RW и RO) и размещаются внутри группы в алфавитном порядке имен секций	T	/—SG
/SS	В файл глобальных символов включаются только определения глобальных символов, на которые имеются ссылки	I	/—SS
/TR	Задача строится с установленным битом слежения T в PSW, при выполнении первой инструкции задачи произойдет синхронное системное прерывание по биту слежения	T	/—TR

Продолжение

Формат	Значение	Тип файла	Значение по умолчанию
/WI	Указывает широкий формат (132 символа) распечатки файла распределения памяти, при отрицании ключа ширина формата устанавливается равной 80 символам	M	/WI
XT[:n]	ТКВ завершает построение задачи после обнаружения n-ошибок	T	/—XT n=1

5.7. НЕОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ

Необязательные параметры используются для указания характеристик строящейся задачи. Построитель задач запрашивает ввод необязательных параметров, если за спецификацией вводных файлов указывается знак «/». Необязательный параметр имеет формат:

ключевое слово=список аргументов

Аргументы в списке разделяются знаком двоеточия. В одной строке может быть задано несколько необязательных параметров, разделенных знаком «!».

Некоторые параметры имеют несколько списков аргументов, для разделения этих списков используется запятая.

Пример:

```
TASK=NAME ! PRI=100
COMMON=DATA : RW ! LIBR : RO
```

Далее приводится формат необязательных параметров и их назначение.

Параметр ABORT обеспечивает перезапуск ТКВ без создания выводных файлов. Используется при ошибке, допущенной ранее, в последовательности ввода командных строк или необязательных параметров. Формат:

```
ABORT=n
```

где *n* — любое число, значение которого игнорируется.

Параметр ABSPAT обеспечивает до восьми коррекций для указанного сегмента. Формат:

```
ABSPAT=имя сегмента: адрес: значение 1: значение 2...
```

где — имя сегмента содержит от 1 до 6 символов; адрес — восьмеричный адрес первой коррекции; значение 1, 2 ... — вось-

меричные числа в пределах от 0 до 177777, записываемые в последовательные ячейки относительно заданного адреса.

Параметр ACTFIL объявляет число файлов, которое задача может иметь одновременно открытыми. Для каждого открытого файла выделяется около 512 байт. Параметр используется только для программ на Фортране. Формат:

ACTFIL=n

где *n* — число файлов.

Значение по умолчанию: ACTFIL=4.

Параметр ASG назначает допустимые логические номера для одного или нескольких устройств. Формат:

ASG=имя устройства: номер1: номер 2: ... номер 8

Значение по умолчанию:

ASG=SY0:1:2:3:4, TI0:5, CL0:6

Параметр CLSTR объявляет группу системных разделяемых областей (от двух до шести), с которыми связывается задача, и имеет к ним доступ с использованием общего разделяемого виртуального адресного пространства. Разделяемые общие области данных или разделяемые библиотеки и соответствующие им файлы определения символов должны быть размещены на устройстве *LB*: в каталоге [1,1]. Формат:

CLSTR=имя 1, имя 2, ..., имя N: ключ[:регистр]

где имя 1, имя 2, ... — имена разделяемых областей; в группе может быть задано от двух до шести областей. Первая разделяемая область отображается при активизации задачи, другие области переотображаются в общее разделяемое виртуальное адресное пространство при обращении к ним;

ключ определяет общий режим доступа ко всем разделяемым областям группы, *RW* — чтение и запись, *RO* — только чтение;

регистр определяет номер (от 1 до 7) первой пары регистров ДП, используемых для отображения разделяемых областей и указывается только для позиционно-независимых областей.

Параметры COMMON и *LIBR* являются идентичными и объявляют, что задача использует системную разделяемую область (соответственно системную разделяемую область данных или системную разделяемую библиотеку). Формат:

COMMON=имя: код доступа [: регистр]
LIBR =имя: код доступа [: регистр]

где имя — имя системной разделяемой области, файл определения символов для разделяемой области должен находиться на устройстве LB0: в каталоге [1,1];

код доступа — RW или RO;

регистр определяет номер (от 1 до 7) первой пары регистров ДП для отображения разделяемой области и указывается только для позиционно-независимых областей.

Параметр EXTSCT объявляет расширение размера программной секции. Для программной секции с атрибутом CON секция расширяется на указанное число байтов; для секции с атрибутом OVR размер программной секции устанавливается равным длине расширения, если длина расширения больше длины секции. Формат:

EXTSCT=имя программной секции: n

где *n* — восьмеричное число, определяющее расширение программной секции в байтах.

Параметр EXTTSK объявляет выделение дополнительной памяти для задачи, устанавливаемой в системно-управляемый раздел. Формат:

EXTTSK=n

где *n* — десятичное число слов приращения задачи.

Параметр FMTBUF определяет размер рабочего буфера, выделяемого для компиляции спецификаций формата, вводимых во время выполнения. Параметр используется только для программ на Фортране. Формат:

FMTBUF=n

где *n* — десятичное число, определяемое количеством символов самой длиной спецификации. По умолчанию n=72.

Параметр GBLDEF объявляет определение глобального символа, отвергая определение этого символа во всех модулях. Формат:

GBLDEF=имя глобального символа: n

где n — восьмеричное число в пределах от 0 до 177777.

Параметр GBLINC указывает ТКВ о включении определения одного или нескольких глобальных символов, заданных в параметре, в файл определения символов. Формат:

GBLINC=имя 1, имя 2, ... имя N

где имя 1 ... имя *N* — имена включаемых глобальных символов.

Параметр GBLPAT обеспечивает возможность до восьми корректировок относительно глобального символа. Формат:

GBLPAT=имя сегмента: символ 1±смещение 1: n1: n2: ... , n8

где смещение — восьмеричное число, определяющее смещение относительно глобального символа;

n1, n2,...,n8 — восьмеричные числа, записываемые последовательно в слова с заданным смещением относительно глобального символа.

Параметр GBLREF объявляет ссылку на глобальный символ. Ссылка порождается в корневом сегменте. Формат:

GBLREF=имя глобального символа

Параметр GBLXCL обеспечивает возможность исключения указанных глобальных символов из файла определения символов. Формат:

GBLXCL=символ 1, символ 2, ...

Параметр MAXBUF объявляет максимальный размер буфера записей для файлов, обрабатываемых задачей. Параметр используется только для программ на Фортране. Формат:

MAXBUF=n

где n — десятичное число, определяющее максимальный размер записи в байтах.

Параметр ODTV объявляет указанный глобальный символ адресом таблицы векторов SST — прерываний отладчика. Глобальный символ должен быть определен в корневом сегменте. Формат:

ODTV=символ: n

где n — длина таблицы векторов SST-прерываний в словах.

Параметр PAR определяет раздел, для которого строится задача. В системах с ДП задачу можно установить в любом разделе — управляемом системой или пользователем. В системах без ДП задача при построении связывается с базовым адресом раздела и может быть установлена только в разделе, для которого она построена, или в разделе с другим именем, имеющим такой же базовый адрес. Формат:

PAR=имя раздела[: база: длина]

где база — восьмеричный базовый адрес раздела; в системе без ДП база является физическим адресом; в системе с ДП для задач база должна быть нулевой, а для разделяемых областей — кратной 4 Кловам;

длина — восьмеричное число, определяющее длину раздела; в системе с ДП нулевая длина определяет раздел, управляемый системой, а ненулевая длина — управляемый пользователем.

Значение по умолчанию: PAR=GEN.

Параметр PRI определяет приоритет выполнения задачи. В системе с многопользовательской защитой задача,

запускаемая с непривилегированного терминала, не может иметь приоритет выше 50. Формат:

PRI=приоритет

где приоритет — десятичное число от 1 до 250.

Параметры RESCOM и *RESLIB* объявляют, что задача использует резидентную разделяемую область (соответственно резидентную разделяемую область данных или резидентную библиотеку). Формат:

RESCOM=файл/код доступа [:регистр]
RESLIB=файл/код доступа [:регистр]

где файл — спецификация файла образа резидентной области данных или резидентной библиотеки, файл таблицы символов должен находиться в том же каталоге, в каком находится файл образа;
код доступа — RW или RO;
регистр определяет номер (от 1 до 7) первой пары регистров ДП для отображения разделяемой резидентной области и может указываться только для позиционно-независимых разделяемых областей.

Параметр STACK объявляет размер стека задачи. Формат:

STACK=n

где n — десятичное число, определяющее размер стека в словах.

Значение по умолчанию STACK=256.

Параметр TASK определяет имя задачи. Формат:

TASK=имя задачи

Параметр TSKV объявляет глобальный символ адресом таблицы векторов SST-прерываний задачи. Указанный символ должен быть определен в главном корневом сегменте. Формат:

TSKV=символ: n

где n — длина таблицы векторов SST-прерываний в словах.

Параметр UIC объявляет код идентификации задачи при ее выполнении. Формат:

UIC= [гр, чл.]

Значение по умолчанию: код идентификации, под которым выполняется ТКВ (обычно *UIC* терминала).

Параметр UNITS объявляет число логических номеров устройств, используемых задачей. Формат:

UNITS=n

где *n* — число в пределах от 0 до 250, указывающее максимальное число логических номеров устройств, используемых задачей, а также максимальный допустимый логический номер устройства.

Значение по умолчанию: UNITS=6.

Параметр VSECT обеспечивает распределение виртуальной памяти для именованной программной секции, что позволяет создавать большие структуры данных и ссылаться к ним с помощью директив управления памятью. Формат:

VSECT=имя секции: база: окно [:район]

где база — виртуальный базовый адрес программной секции, кратный 4 Кловам;

окно — размер виртуального адресного пространства, выделяемого программной секции; сумма базы и окна не должна превышать значение 177777;

район — размер физической памяти в 32-словных блоках, выделяемой в районе задачи для размещения программной секции. Секция может отображаться в эту физическую память при выполнении задачи. Параметр «район» является необязательным, при его умолчании предполагается, что район для отображения программной секции будет создаваться при выполнении задачи по директиве GRRC⊙

Параметр WNDWS объявляет дополнительное число адресных окон, необходимых задаче, кроме окон, используемых для отображения образа задачи и разделяемых областей. Формат:

WNDWS=n

где n — число от 0 до 7.

Значение по умолчанию: WNDWS=0.

6

СИСТЕМНЫЕ ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ

6.1. ОБЩИЕ СВЕДЕНИЯ ОБ ОБСЛУЖИВАЮЩИХ ПРОГРАММАХ ОСРВ

Для широкого круга пользователей СМ ЭВМ качество операционной системы во многом определяется ее сервисными средствами. В ОСРВ такими средствами являются удобные и достаточно простые обслуживающие программы (утилиты), рассчитанные на программистов и операторов различной квалификации. С помощью этих программ создают, преобразовывают и редактируют файлы, переписывают их с носителя на носитель, сохраняют и восстанавливают тома, отлаживают программы, обрабатывают библиотеки и т. д. Обслуживающие программы ОСРВ условно можно разделить на следующие классы:

программы обработки файлов;
программы проверки и подготовки носителей;
программы сохранения и восстановления томов;
отладочные программы;
библиотекарь;
программы редактирования*.

Работа с различными обслуживающими программами имеет ряд общих черт, к которым можно отнести:

формат командной строки;
спецификации вводных и выводных файлов;
способы запуска обслуживающих программ.

---

* Программы редактирования служат для создания и корректировки символьных файлов. В состав утилит ОСРВ входит несколько программ-редакторов: текстовой, экранный, универсальный, программируемый и пакетный. Работа с каждым из них достаточно проста, однако описание их возможностей и набора команд так велико, что выходит за пределы, допускаемые объемом данного справочника.

Формат командных строк для обслуживающих программ:

выв ф1 [, ... , выв фN] = вв ф1 [, ... , вв фN]

В зависимости от применяемой программы формат командной строки может быть изменен. Так, большинство обслуживающих программ используют косвенные командные файлы, а некоторые программы — сокращенную форму командной строки.

Число спецификаций файлов в командных строках зависит от конкретной программы, но общая длина строки ограничена. Каждая спецификация файла имеет стандартный формат, принятый в ОСРВ:

устр: [гр, чл] имя файла. тип ; версия /ключ, ... , / подключ

Подробно все компоненты спецификации файла разбираются в гл. 1 и 3.

Как правило, функции обслуживающих программ задаются с помощью ключей и подключей. Знак «—» и символы NO отвергают действие ключа. Ключи могут иметь аргументы в символьном и цифровом виде. Цифровые значения аргументов по умолчанию восьмеричные. Для указания десятичного числа его необходимо ограничить десятичной точкой. Подключ — двухсимвольное имя — уточняет действие ключа; синтаксически правила ввода ключей и подключей аналогичны.

Пример:

```
PIP> [200,200]*.*;*/PR/FO
подключ /FO используется совместно с ключом /PR
```

Обслуживающие программы в ОСРВ начинают работу по командам оператора или через косвенный командный файл. Обслуживающие программы запускаются в ответ на подсказку программы связи с оператором (MCR). Установленные программы запускаются двумя способами:

```
> имя-программы командная-строка
или
> имя-программы
```

где > — подсказка MCR.

При первом способе запуска программа выполняет требуемые действия и завершается, а при втором — командная строка может вводиться многократно, до тех пор пока оператор не завершит обслуживающую программу управляющим символом CTRL/Z.

Неустановленные обслуживающие программы запускаются с помощью команды MCR RUN. При этом командная строка имеет формат:

RUN ⊙имя программы

После получения программой управления она печатает подсказку

имя обслуживающей программы>

и ждет ввода командной строки.

Если обслуживающей программе нужно работать с кодом UIC, отличным от текущего, то в команде RUN указывается ключ /UIC=[гр., чл.]. Так же как и при втором способе запуска установленных программ, командная строка может вводиться оператором многократно. Для большинства обслуживающих программ командные строки могут вводиться и из косвенных командных файлов.

6.2. ПРОГРАММЫ ОБРАБОТКИ ФАЙЛОВ

Программы обработки файлов служат для переписи, преобразования, удаления, сравнения файлов, а также для выполнения других вспомогательных функций, необходимых пользователям при работе с файлами.

6.2.1. ПРОГРАММА PIP

Программа работы с файлами PIP служит для переписи файлов с одного устройства с файловой структурой ОСРВ на другое, а также для копирования, удаления, переименования, разблокирования файлов и распечатки каталогов. Формат командной строки PIP может меняться в зависимости от выполняемых функций. При задании в строке нескольких команд они отделяются друг от друга знаком &.

Умолчания для спецификаций файлов PIP соответствуют общепринятым, но у программы PIP существует ключ /DF, явно определяющий умолчания. При употреблении нескольких файлов в командной строке умолчания в спецификациях последующих файлов соответствуют спецификациям в предыдущих.

Ключи команды глобальные и могут быть указаны один раз для всего списка файлов с любой стороны от знака равенства.

Употребление специальных символов. Программа PIP разрешает применение символа * в одном или нескольких полях спецификации файла. Символ * используется вместо явного задания поля и означает соответственно все файлы, все типы и т. д. Знак % может указываться в поле имени и типа файла. В этих полях знак % означает, что на его месте может быть любой допустимый символ.

Символ * не может использоваться в спецификациях выводных файлов в следующих случаях:

при конкатенации файлов с указанным файлом;

в дополнение к существующим файлам;

при корректировке существующего файла;

при распечатке каталога.

Ключи программы работы с файлами. Отсутствие в командной строке ключей и подключей определяет копирование файла на то же самое или другое устройство или слияние нескольких вводных файлов в выводной файл. Ниже перечисляются ключи программы PIP и дается краткая характеристика их функционирования.

/AP — открывает существующий файл и добавляет файлы ввода в конец существующего файла. Формат:

```
выв ф = вв ф1 [, вв ф2, . . , вв фN] /AP/FO
```

где /FO — подключ принадлежности файла, который указывает, что код UIC владельца выводного файла должен соответствовать коду выводного каталога. Если подключ /FO не указан, то UIC нового файла будет тот, под которым работает программа PIP.

/BS — служит для определения размера блока магнитной ленты. Формат:

```
выв ф /BS:N[·] = вв ф
```

где N — размер блока в байтах.

/CD — используется для копирования файлов, при этом дата создания нового файла берется от старого. Формат:

```
выв ф/CD = вв ф1 [, вв ф2, ... , вв фN]
```

/DD — осуществляет поиск файлов, которые были созданы в указанный период времени между начальной и конечной датой. Формат:

```
/DD· начальная дата: конечная дата
```

Вместо начальной или конечной даты можно употреблять символ *.

Пример написания даты: 05—JUN—84 (5 июня 1984 года).

/DE — служит для удаления файлов. Задав подключ /LD, можно распечатать на терминале имена удаляемых файлов. При удалении файлов умолчание в поле версии недопустимо. Формат:

вв ф1 [,ввф2, ...,вв ф N]./DE.[/LD]

/DF — позволяет установить умолчание для устройства или кода UIC в спецификациях файлов. Формат:

устр: [UIC]/DF, или устр: /DF, или [UIC]/DF, или /DF

В последнем случае по умолчанию устанавливается устройство SY∅ и UIC, под которым работает программа PIP.

/EN — обеспечивает ввод синонима для файла, разрешая таким образом обращение к файлу с помощью нескольких имен. Формат:

выв ф=вв ф1 [, вв ф2, ..., вв ф N]/EN[/NV]

где /NV — подключ новой версии.

В спецификации выводного файла имя, тип и версия могут задаваться явно, символом * или быть пустыми. Наличие умолчания или символа * означает, что должны использоваться соответствующие поля вводного файла.

Для выводного и вводных файлов нельзя указывать разные устройства.

/EOF — позволяет записать признак конца файла на указанное место, в результате сохраняется часть разрушенного файла. Формат:

вв ф/EOF [:блок:байт] [,..., вв ф/EOF [:блок:байт]]

где блок — номер блока, в котором устанавливается признак конца файла; байт — номер байта (не более 512), в котором определяется признак конца файла. Если номера блока и байта не указаны, то признак конца файла устанавливается в последний байт последнего блока непрерывного файла.

/EX — удаляет из процесса поиска указанные файлы. Из поиска можно исключить целые классы имен, типов и версий, указывая в соответствующих полях символ *; в спецификации нельзя указывать *. *; *, нельзя также указывать только устройства и каталоги. Заданный ключ /EX действует далее по умолчанию. Формат:

спецификация файла /EX

/FI — разрешает доступ к файлу по его идентификатору. Формат:

выв ф = /FI: номф : послномф

где номф — номер файла;
послномф — последовательный номер файла (см. гл. 3).

/FR — распечатывает общее число свободных блоков на устройстве, наибольший размер непрерывного пространства, количество заголовков файла и количество занятых заголовков файла. Формат:

устр /FR

/ID — выдает на терминал номер версии программы PIP и сообщает о возможности работы с магнитной лентой.

/LI, /BR, /FU, /TB — ключи распечатки каталога; выводят один или более каталогов в различных форматах. Формат:

[файл листинга] = [ввф1] [, ввф2, ..., ввфN] /ключ [/подключ]

Умолчание для файла листинга — TI:, умолчание для вводного файла *.*;*. По ключу /BR дается краткая распечатка каталога — имя файла, тип, версия. По ключу /LI в дополнение к этой информации распечатывается:

число использованных блоков (десятичное);

код файла: пробел — прерывистый, C — непрерывный, *L* — блокированный;

дата и время создания файла;

завершающая строка, которая содержит число использованных блоков, число зарезервированных блоков и общее число файлов в каталоге (все числа десятичные).

/FU [:N] — по этому ключу распечатывается каталог в полном формате, N — указывает число символов в строке. В дополнение к информации, печатаемой по ключу /LI, выводится:

идентификатор файла в формате: номер файла, последовательный номер файла;

код идентификации владельца файла ([гр, чл.]);

код защиты файла в формате «системная», «владелец», «групповая», «общая» (см. ниже ключ /PR). Каждое поле защиты содержит комбинацию значений

R, W, E, D

где R — чтение разрешено, W — запись разрешена, E — расширение разрешено, D — удаление разрешено;

дата и время последней корректировки и число корректировок.

/NM — позволяет не печатать некоторые сообщения об ошибках. Формат:

ввф1 [,ввф2, ... , ввфN] [/ключ]/NM

где ключ — некоторые комбинации ключей и подключей (LI, DE, PU и т. д.).

/PR — позволяет изменять привилегии доступа к файлу. Защита обеспечивается для четырех категорий пользователей: «системная», «владельца», «групповая», «общая». Для каждой категории может быть указано право на чтение, запись, расширение и удаление файла. Формат:

ввф/PR[/SY[:RWED]] [/OW[:RWED]] [/GR[:RWED]]
[/WO[:RWED]] [/FO]

где спецификация ввф должна быть задана явно;

/SY — подключ системной категории;

/OW — подключ собственной категории;

/GR — подключ групповой категории;

/WO — подключ общей категории. С помощью этих подключей пользователь указывает, какая из категорий защиты должна быть изменена;

значения RWED имеют тот же смысл, что и в ключе /FU и указывают, какие права доступа допустимы для соответствующей категории пользователей;

/FO — подключ принадлежности файла владельцу, он устанавливает код идентификации владельца файла, равный коду UIC каталога, в котором файл указан.

/PU — обеспечивает возможность удаления устаревших версий файла. Формат:

ввф1 [, ввф2, ... , ввфN] /PU:N[/LD]

Если указано значение N и последняя версия файла имеет номер M, то версии с номерами, меньшими или равными M—N, удаляются. Если N не указано, удаляются все версии файла кроме последней. Значение N применимо к спецификации файла, непосредственно предшествующей ему (локально). При задании подключа /LD имена удаляемых файлов будут распечатываться на терминале.

/RE — обеспечивает возможность изменить имя и тип файла. Формат:

вывф=ввф1 [, ввф2, ... , ввфN] /RE[/NV]

С помощью подключа /NV пользователь может увеличить на один номер версию переименованного файла по сравнению с последней версией ранее существовавшего файла.

Переименование файлов должно производиться на одном устройстве.

/RM — позволяет убирать запись из каталога.

В отличие от /DE этот ключ не стирает файл, а удаляет только запись из каталога. Формат:

ввф1 [, ввф2, ... , ввфN] /RM

/RW — служит для перемотки магнитной ленты. Формат:

вывф [/RW] = ввф [/RW]

В том случае, когда ключ /RW задан со спецификацией выводного файла, лента перематывается на начало, когда же ключ задан со спецификацией вводного файла, лента позиционируется перед этим файлом.

/SB — при переписи файла с магнитной ленты на диск позволяет пересекать записям границы блоков. Формат:

вывф/SB = ввф

/SD — в процессе диалога с оператором удаляет или оставляет файлы в каталоге. Формат:

ввф1 [, ввф2, ... , ввфN] /SD

После ввода команды программа PIP для каждого файла запрашивает ответ:

DELETE FILE ввф [Y/N/G/Q]

Оператор должен ответить, нажав клавишу с одной из перечисленных букв:

Y — удалить файл и продолжить просмотр;

N — сохранить файл и продолжить просмотр;

Q — сохранить файл и завершить просмотр;

G — удалить все версии файла и завершить просмотр.

Если в комбинации с этими буквами оператор набирает символы CTRL/Z, то после выполнения заданной функции происходит выход из программы PIP.

/SP — задает список файлов, выводимых на печать асинхронно с помощью задачи системного вывода. Формат:

вывф = [, ввф2, ... , ввфN]

где вывф — спецификация файла, выводимого на печать;
N — число копий вывода.

/SR — позволяет прочитать файл, который был открыт для записи другой задачей. Формат:

вывф = ввф/SR

/TD — осуществляет поиск только тех файлов, которые были созданы за текущий день. Формат:

/TD

/TR — позволяет усекать конец файла до логического признака конца. Формат:

ввф1 [, ввф2, ... , ввфN] /TR

/UF — используется для создания каталога файлов на томе. При включении файла в каталог должна быть указана принадлежность файла (подключ/FO). Формат:

вывф [/UF [/FO]] = ввф1, ... , ввфN

/UN — позволяет разблокировать файл, который был заблокирован в результате некорректного закрытия. Формат:

ввф1 [, ввф2, ... , ввфN] /UN

/UP — позволяет записывать в существующий файл новые данные от его начала. Формат:

вывф = ввф1 [, ввф2, ... , ввфN] /UP [/FO]

где /FO — подключ принадлежности файла.

6.2.2. ПРОГРАММА ПРЕОБРАЗОВАНИЯ ФАЙЛОВ *FLX*

Программа FLX производит файловые преобразования из форматов систем ДОС СМ и РАФОС в форматы файловой структуры ОСРВ и наоборот:

из ДОС СМ в ОСРВ и наоборот из ОСРВ в ДОС СМ;

из РАФОС (ФОБОС) в ОСРВ и наоборот из ОСРВ в РАФОС.

Кроме того, программа преобразования файлов выполняет дополнительные функции:

распечатывает каталоги томов и кассет РАФОС и ДОС СМ;

удаляет файлы из томов ДОС СМ и РАФОС;

инициирует тома ДОС СМ и РАФОС.

Программа FLX требует не менее 8 Кслов.

Командная строка FLX состоит из необязательной спецификации выводного файла и нескольких спецификаций вводного файла в следующем формате:

вывф/ключ = ввф1/ключ [, ввф2/ключ, ... , ввфN/ключ]

Допустимый формат спецификаций выводного файла:

устр: [гр, чл].

Символ * употребляется только в вводных файлах,

номера версий используются только для файлов ОСРВ и не могут заменяться символом *.

Ключи программы преобразования файлов. FLX работает с тремя типами ключей для файловых передач.

1. Ключи типа формата определяют формат файла: /DO идентифицирует файл в формате ДОС СМ, /RS — в формате ОСРВ, /RT — в формате РАФОС.

По умолчанию предполагается ключ /DO для вводных и ключ /RS для выводных файлов. Чтобы изменить умолчания на противоположные, необходимо задать ключ FLX> /RS.

Пример:

```
FLX> DP1:=DK3: FILE1.MAC
```

В этом примере файл FILE1.MAC переписывается с тома ДОС СМ (DK3:) на том ОСРВ (DP1:).

2. Ключи типа передач управляют форматом передачи: двоичным, символьным, форматом образа.

/FA:N — форматный символьный ключ. Символьные файлы ДОС и РАФОС должны иметь форматный тип. Он определяет записи в семибитном символьном коде, заканчивающиеся символами возврата каретки и перевода строки (CR—LF), перевода формата (FF), вертикальной табуляции (VT). В передачах файлов из ДОС или РАФОС в ОСРВ пары CR—LF удаляются в концах записей. В передачах из ОСРВ в ДОС или РАФОС пары CR—LF добавляются в конец записей, не имеющих в конце символов LF или FF. N — размер записи. Если N не указано для выводного файла ОСРВ, то генерируются записи переменной длины.

/FB:N — форматный двоичный ключ. Двоичные файлы ДОС и РАФОС должны быть форматными. В этом случае форматные двоичные заголовки и контрольные суммы добавляются при выводе записей в файлы ДОС или РАФОС, и, наоборот, они убираются при передаче файлов в ОСРВ.

/IM:N — ключ формата образа. При передаче в формате образа создаются непрерывные файлы с записями фиксированной длины (по умолчанию N=512 байт).

Подразумеваются также типы файлов по умолчанию:

Ключ	Тип файлов
/IM	.TSK, .OLB, .MLB, .SYS, .SML, .ULB, .EXE
/FB	OBJ, .STB, .BIN, .LDA
/FA	все другие

3. Ключи управления файлом служат для управления обработкой файлов.

/BL:N — указывает число смежных блоков, которые должны быть выделены для размещения выводного файла; N — число блоков.

/CO — указывает, что выводной файл должен быть непрерывным (имеет смысл только для дисков).

/DE — в комбинации с ключами /DO или /RT используется для удаления файлов с томов ДОС СМ или РАФОС.

/DI(/LI) — в комбинации с ключами /DO или /RT вызывает распечатку каталогов томов ДОС СМ или РАФОС.

/DNS:N — определяет плотность записи на магнитной ленте типа ММ:. При плотности 32 бит/мм необходимо указать N=800, а при плотности 64 бит/мм — 1600.

/FO — указывает, что должны быть использованы соглашения относительно управления кареткой, принятые в языке Фортран.

/ID — выдает номер текущей версии программы FLX.

/NU:N — используется, чтобы указать число блоков каталога N при инициализации диска РАФОС. (Используется вместе с ключами /RT и /ZE).

/RW — служит для перемотки магнитной ленты на начало.

/SP — указывает, что выводной файл должен обрабатываться задачей системного вывода (только для ОСРВ).

/UI — указывает, что выводной файл должен иметь тот же UIC, что и вводной.

/VE — предназначен для чтения и проверки каждой записи, выводимой на кассетную магнитную ленту.

/BS:N — указывает размер блока при выводе на кассетную магнитную ленту (по умолчанию 128 байт).

/ZE[:N] — инициирует тома ДОС или РАФОС; N — указывает количество блоков каталога.

6.2.3. ПРОГРАММА СРАВНЕНИЯ ФАЙЛОВ СМР

Программа СМР построчно сравнивает в символьном коде два файла для того, чтобы определить, являются ли параллельные записи идентичными.

Программа сравнения файлов выводит:

листинг, в котором указаны различия между файлами.

Слева печатаются строки первого файла, у которых есть различия, справа — строки второго файла;

листинг файла с различиями, отмеченными восклицательными знаками (!);

командные строки для программы SLP (пакетный редактор). Эти строки содержат команды SLP, с помощью которых можно сделать первый файл идентичным второму.

Формат командной строки CMP:

```
вывф/ключ=ввф1, ввф2
```

где вывф — файл листинга;
ввф1 и ввф2 — сравниваемые файлы.

Ключи программы сравнения файлов. Ниже перечисляются ключи программы CMP и дается краткая характеристика их функционирования.

/BL — указывает, что пустые строки в обоих вводных файлах включаются в процесс сравнения. По умолчанию / — BL.

/CB — определяет, что программа CMP распечатывает ввф2, в котором каждая строка, не идентичная строке ввф1, отмечается знаком !. По умолчанию /—CB.

/CO — указывает, что комментарии включаются в процесс сравнения.

/DI — вызывает распечатки всех имеющихся различий в двух файлах.

/FF — указывает, что записи, состоящие из символа FF (перевод формата), включаются в процесс сравнения.

/LI:N — определяет число строк (N) в обоих файлах, которые должны совпадать, чтобы программа CMP опознала конец несовпадения (по умолчанию 3).

/LN — показывает, что строкам в выводном файле предшествуют номера.

/MB — указывает, что все пробелы и знаки табуляции сравниваются.

/SL — по этому ключу программа CMP создает выводной файл для программы SLP.

/SP[:N] — напоминает, что выводной файл будет выводиться программой системного вывода (N — количество копий файла).

/TB — указывает, что все пробелы в конце строки сравниваются.

/VB:NNN — определяет восьмеричный символьный код, используемый как признак несовпадения.

6.2.4. ПРОГРАММА РАСПЕЧАТКИ ФАЙЛОВ *DMP*

Программа распечатки файлов DMP выводит содержимое томов и файлов на устройства вывода в различных форматах. Программа DMP работает в файловом режиме или в режиме устройства. Можно распечатывать заголовки и виртуальные блоки файла или только записи.

При распечатке тома может быть распределена область распечатываемых логических блоков.

В файловом режиме определяется один вводной файл, который распечатывается весь или частично по указанию области виртуальных блоков. При этом вводной том должен быть монтирован. Данные распечатываются записями или блоками.

В режиме устройства указывается немонтированное устройство ввода и распечатывается вся область логических блоков, заданная ключом /BL:N:M.

Запуск программы DMP производится так, как описано в разд. 6.1.

Формат командной строки DMP:

[выв.файл] [/ключ] [/ключ ...] = вв.файл [/ключ] [/ключ ...]

Если спецификация выводного файла отсутствует, создается файл DMPFIL.DMP.

Ключи программы *DMP*. Ниже перечисляются ключи этой программы и дается краткая характеристика их функционирования.

/AS — указывает, что данные будут распечатаны в символьном виде.

/BA:N:M — позволяет определять двухсловный базовый адрес блока, где N и M восьмеричные. Этот ключ используют для блоков, номера которых превышают 16 байт, при этом номера блоков, заданные ключом /BL, будут прибавляться к базовому адресу для получения действительного значения адреса блока. Значение 1 для базового адреса соответствует 200000_8.

/BL:N:M — указывает область выводимых блоков (N — первый, M — последний).

/BY — указывает, что данные выводятся в байтном, восьмеричном формате.

/DC — указывает, что данные выводятся в формате десятичных слов.

/DENS:N — определяет плотность записи на магнитной ленте (ММ:). При N=800 плотность равна 32 бит/мм, а при N=1600 64 бит/мм.

/FI:НФ:ПН — использует номер файла (НФ) и последовательный номер файла (ПН) как идентификатор вводного файла, используемый вместо имени.

/HD[:F:U] — употребляется в файловом режиме для распечатки заголовка файла.

При указании :F (по умолчанию) заголовок распечатывается в формате файловой структуры ОСРВ, при указании :U заголовок распечатывается в неформатированном восьмеричном виде.

/HF — форматирует только блоки данных, являющиеся заголовками файлов ОСРВ.

/HX — указывает, что данные выводятся в шестнадцатеричном байтовом формате.

/ID — идентифицирует версию программы DMP.

/LB — выводит номер логического блока, с которого размещается на устройстве файл.

/LC — распечатывает данные прописными буквами русского алфавита.

/LW — выводит данные в формате двойных шестнадцатеричных слов.

/MD[:N] — управляет при распечатке номерами строк. Этот ключ позволяет не сбрасывать в ноль значение номера строк на границе блока, (N) задает значение первой строки (по умолчанию 0).

/OCT — распечатывает данные в восьмеричном формате, не подавляя другие заданные форматы, кроме /AS.

/R5 — выводит данные в формате RADIX-50.

/RC — указывает, что данные выводятся записями, а не поблочно.

/RW — перематывает ленту к началу перед обращением к ней.

/SB:N, где N — указывает число блоков, которое нужно пропустить вперед (N) или назад (-N) на ленте.

/SF:N, где N — указывает число признаков EOF, которое нужно отсчитать при перемотке ленты вперед (N) или назад (-N).

/SP — указывает, что файл выводится на печать с помощью задачи системного вывода.

6.3. ПРОГРАММЫ ПРОВЕРКИ И ПОДГОТОВКИ НОСИТЕЛЕЙ

6.3.1. ПРОГРАММА ФОРМАТИРОВАНИЯ ДИСКА FMT

Программа служит для форматирования и проверки дисковых томов типа DK:, DM:, DP: и DY:.

Программа выполняет следующие функции:

пишет полный заголовок для каждого сектора на диске при форматировании;

проверяет содержимое адреса заголовков каждого сектора;

устанавливает плотность записи на диске DY:;

задает максимально допустимое число ошибок для диска;

запускает программу BAD через директивы порождения задач.

Запуск программы FMT производится по команде MCR. Формат командной строки:

```
FMT DDN: [/ключ1 .../ключN]
```

где DDN — мнемоника и номер форматируемого тома.

Ключи программы FMT. Ниже перечисляются ключи этой программы и дается краткая характеристика их функционирования.

/BAD — вызывает задачу обнаружения дефектных блоков.

/DENS=(HIGH, LOW) — устанавливает высокую или малую плотность записи на дискете (DY:), по умолчанию — низкая плотность (LOW).

/ERL — задает предельное число ошибок для форматируемого тома, при достижении этого числа программа FMT выдает сообщение и заканчивает работу.

/MAN — служит для ручного режима работы FMT и позволяет форматировать отдельный сектор или дорожку диска.

/NOVERIEFY — отменяет операции по проверке чтения.

/OVR — заставляет игнорировать блок описателя дефектов (последний), записанный при изготовлении диска (см. этот же ключ в программе BAD).

/VE — запускает операции чтения для проверки корректности записей заголовков.

/WLT:N — перезаписывает блок описателя дефектов на диске DM: после того, как программа FMT доба-

вит обнаруженные дефектные секторы к имеющимся в блоке описателя дефектов. Десятичное число N используется как порядковый номер тома.

/@Y — требует ввода команд для программы FMT из косвенного командного файла.

6.3.2. ПРОГРАММА ПОИСКА ДЕФЕКТНЫХ БЛОКОВ *BAD*

Программа BAD проверяет дисковые пакеты для определения числа и места дефектных блоков на носителе. Эта информация записывается в пригодный для использования блок на диске и при инициации тома дефектные блоки отмечаются как занятые. Команда MCR INI распределяет их в файл [0,0] BADBLK.SYS.

Командная строка для вызова программы BAD имеет следующий формат:

```
BAD DDUU: [/ключ ...]
```

где DD — имя устройства;
UU — номер устройства.

Все ключи для устройства должны быть заданы в одной строке. Перед запуском программы BAD диск должен быть сформирован.

Ключи программы BAD. Ниже перечисляются ключи этой программы и дается краткая характеристика их функционирования.

/ALO: метка тома — запрашивает ввод дополнительных дефектных блоков. При использовании этого ключа нет необходимости в повторной инициализации диска, работа ведется с монтированным устройством и пользователь должен быть привилегированным.

/LI — распечатывает на терминал номера всех дефектных блоков, которые были обнаружены во время проверки устройства.

/MAN — запрашивает ввод информации о дефектных блоках, выдавая подсказку на терминал, а затем проверяет диск, включая в блок описателей дефектных блоков информацию, введенную с терминала.

/OVR — игнорирует информацию на последней дорожке и блок описателя дефектных блоков записывается в последний пригодный для использования блок перед последней дорожкой, т. е. диск DM: (с последней дорожкой) рассматривается как устройство, не имеющее ее.

/PAT=M:N — служит для поиска дефектных блоков, с помощью двухсловных тестовых данных, вводимых пользователем как числа M и N (шестнадцатибитные восьмеричные). Стандартно используемые тестовые данные 165555 и 133333.

/RETRY — служит для корректировки ошибок оборудования средствами драйвера. Следовательно, устранимые ошибки, такие, как ошибки контрольной суммы, могут быть устранены, и блок будет отмечен как годный для использования.

/UPD — служит для считывания блока описателя дефектных блоков и выдачи подсказки на ввод дополнительных дефектных блоков. При этом BAD не проверяет диск.

Описываемые ниже три ключа работают только с версией программы BAD, ориентированной на работу без операционной системы:

/CSR=NNNNNN, где NNNNNN — новый восьмеричный адрес регистра команд и состояния устройства;

/VEC=NNN, где NNN — новый восьмеричный адрес вектора прерываний устройства. Эти ключи используются при нестандартном подключении устройства;

/WCHK — вызывает контроль записи для каждой записи на устройство.

6.3.3. ПРОГРАММА ПРОВЕРКИ ФАЙЛОВОЙ СТРУКТУРЫ *VFY*

Программа проверки файловой структуры позволяет:

установить возможность чтения тома и корректность файловой структуры на нем;

напечатать число свободных блоков на томе;

найти файлы, которые числятся в индексном файле, но не указаны ни в одном из каталогов;

распечатать все файлы с указанием идентификатора, имени и кода UIC владельца;

отметить как использованные блоки, которые не заняты файлом, но распределены для него;

перестроить файл распределения пространства диска так, чтобы адекватно отразить информацию в индексном файле;

сбросить признак удаления файла;

выполнить проверку чтения каждого блока на томе;

проверить корректность каталогов;

удалить дефектные заголовки файлов.

Во время работы программы VFY на монтированном томе не должны выполняться никакие другие операции.

Формат командной строки программы VFY:

файл листинга, устр. для временного хранения файла=проверяемый том/ключ или проверяемый том/ключ

Временный файл используется программой VFY во время проверки и поиска потерянных файлов. По умол-

чанию создается на устройстве SY∅ и после завершения программы стирается.

Ключи программы *VFY*. Если в команде ключ не указан, то проверяется возможность считывания и корректность файловой структуры тома. При этом считываются все заголовки файла в индексном файле и проверяется, какие дисковые блоки, указанные в области планирования в заголовке файла, отмечены как распределенные.

/DE — позволяет сбросить индикатор удаления в заголовке файла — для тех файлов, которые отмечены для удаления, но не удалены. При этом том монтируется с ключом /UNL и программа VFY выполняется под системным кодом идентификации.

/UP — позволяет отметить как занятые под файлы блоки, которые числятся свободными, но в действительности распределены для файла.

Примечания:

1) файлы с многократно распределенными блоками должны быть удалены до корректировки;

2) временный файл должен находиться на другом томе;

3) программа VFY должна выполняться под системным кодом идентификации;

4) корректируемый том должен быть доступен для записи.

/RE — позволяет восстановить потерянные блоки, которые отмечены как занятые, но никакие файлы их не содержат.

/FR — распечатывает размер доступного свободного пространства на указанном томе.

/LO — дает возможность просмотреть весь том, чтобы отыскать те файлы, которых нет ни в одном каталоге, т. е. потерянные в том смысле, что к ним нельзя обратиться по имени. Если на томе существует каталог [1,3], то эти файлы будут введены в него.

/LI — распечатывает из индексного файла идентификаторы файлов (номер файла, последовательный номер файла, имя и UIC владельца).

/RC[:N] — проверяет, может ли каждый блок каждого файла на томе быть прочитан. :N — это блочный фактор, указывающий число блоков, которые должны быть считаны за один раз.

/DV — проверяет каждый каталог на томе и сообщает о несуществующих файлах, перечисленных в каталоге.

/HD[/AL] — определяет дефектные заголовки файлов на томе. Если задать подключ /AL, все дефектные заголовки файлов будут удалены.

6.4. ПРОГРАММЫ СОХРАНЕНИЯ И ВОССТАНОВЛЕНИЯ ТОМОВ

6.4.1. ПРОГРАММА КОПИРОВАНИЯ И ВОССТАНОВЛЕНИЯ ТОМОВ (*BRU*)

Системная программа BRU позволяет копировать и восстанавливать тома с файловой структурой ОСРВ. Операции копирования и восстановления допустимы для томов, расположенных на магнитных дисках и лентах:

с диска на ленту — операции копирования;

с ленты на диск — операции восстановления;

с диска на диск — операции копирования и восстановления.

Наряду с этими основными функциями программа BRU позволяет:

копировать и восстанавливать выборочные файлы и группы файлов;

инициировать диски до начала операций;

управлять работой лент (перемотка, проверка меток и т. д.);

выдавать вспомогательную информацию и выполнять другие сервисные функции.

Командная строка программы *BRU* отличается от командных строк других утилит и может иметь длину до 256 символов, включая строки продолжения. Основной формат командной строки:

```
BRU> /ключи устр.ввода 1: , ... [спец.ф. , ... ] устр. выв1: , ...
```

Если командная строка имеет вид:

```
BRU> /ключи <CR>
```

то на терминале появляются следующие подсказки:

```
FROM:
TO:
INITIALIZE [Y/N]
```

FROM: — в ответ на эту подсказку вводятся имена устройств, на которых размещены вводные наборы данных в соответствующем формате;

TO: — требует ввода имен выводных устройств в соответствующем формате;

INITIALIZE [Y/N] — вопрос о том, инициализировать ли выводной том.

Если в качестве устройств ввода (вывода) используются магнитные ленты, то может быть задано более одного устройства (MTO:, MT1:). В командной строке может быть указано до 16 спецификаций файлов. Можно

копировать или восстанавливать файлы по именам, каталогам и т. д. (см. п. 6.2.1). Для продолжения командной строки более чем на одну строку в качестве символа продолжения используется дефис (-).

Основные ключи программы *BRU*

/APP[END] — заставляет добавить набор копируемых файлов с вводного тома к последнему из файлов на выводной ленте.

/BAC[KUP SET] — указывает имя копируемого на ленту набора, по умолчанию берется имя дискового тома.

/COM[PARE] — заставляет сравнивать данные на вводном и выводном устройстве и сообщать о всех различиях.

/CRE[ATED]:{BEF[ORE], AFT[ER]}: (дата время) — копирует файлы, созданные до (BEF) или после (AFT) указанной даты. Дата и время (в круглых скобках) вводятся, как в команде TIME MCR (см. п. 4.2).

/DEN[SITY]: значение — задает плотность записи на магнитной ленте. Допустимые значения:

для MT: — 800.=32 бит/мм (по умолчанию);

для MM: — 800.=32 бит/мм (по умолчанию);

для MM: — 1600.=64 бит/мм (по выбору).

/DIR[ECTORY] — распечатывает имена скопированных наборов или файлов на магнитной ленте.

/DIS[PLAY] — распечатывает на терминал имена файлов и их коды UFD.

/ERR[ORS]: число — прекращает операцию восстановления после возникновения указанного числа ошибок при чтении с ленты.

/EXC[LUDE] — исключает из операции копирования или восстановления файлы, заданные в командной строке.

/EXT[END] — задает число блоков, на которое файл может быть расширен при превышении выделенного ему пространства.

/HEA[DERS]: число — задает количество заголовков, под которые резервируется пространство при создании индексного файла.

/INI[TIALIZE] — служит для инициации выводного дискового тома, при этом на томе размещается файловая структура ОСРВ.

/INV[OLUME]: имя — задает метку тома вводного носителя. Имя не может содержать более 12 символов.

/MOU[NTED] — позволяет копировать и восстанавливать файлы на томах, которые монтированы как тома с файловой структурой OCPB.

/NEW[VERSION] — разрешает конфликты в спецификациях файлов, которые возникают во время операции восстановления, созданием новых версий файлов.

/NOI[NITIALIZE] — указывает, что не нужно инициализировать диск, так как он уже имеет файловую структуру OCPB.

/NOS[UPERSEDE] — указывает, что при восстановлении данных на монтированный диск, при идентичности спецификаций файлов, информация с вводного диска не передается.

/OUT[VOLUME]: имя — задает метку тома выводного диска.

/POS[ITION]: {BEGINNING, MIDDLE, END, BLOCK:N} — определяет положение индексного файла на томе при инициализации диска. Параметры BEGINNING, MIDDLE, END задают соответственно начало, середину и конец тома, а параметр BLOCK — номер блока, с которого начинается индексный файл.

/PRO[TECTION]: коды защиты — задает умолчание кодов защиты для файлов, создаваемых на томе после завершения программы BRU (см. ключ PR в п. 6.2.1).

/REV[ISED]:{BEFORE, AFTER}: (дата время) — обеспечивает копирование или восстановление файлов, измененных до или после указанной даты и времени. Аргументы используются такие же, как и в ключе /CRE. Если определяется только дата или время, круглые скобки не нужны.

/REW[IND] — перематывает к началу первую магнитную ленту из последовательности перед выполнением копирования и восстановления.

/SUP[ERSIDE] — позволяет в случае совпадения спецификаций файлов по этому ключу файл на выводном томе заменить файлом с вводного тома.

/TAPE LABEL: метка — задает 6-символьный идентификатор в метке тома на магнитной ленте.

/UFD — создает каталог UFD на монтированном выводном томе.

/VER[IEY] — после копирования данных сравнивает их и сообщает о различиях.

/WIN[DOWS]: число — при инициировании выводного диска задает умолчание для числа указателей извлечения для файла.

6.4.2. ПРОГРАММА СОХРАНЕНИЯ И УПЛОТНЕНИЯ ТОМА *DSC*

Программа сохранения и уплотнения тома копирует файлы, содержащиеся на диске с файловой структурой ОСРВ, на ленту или диск или с ленты формата DSC на диск. Файлы и их расширения размещаются в смежных блоках, что сокращает требуемое число указателей извлечения и заголовков файлов на создаваемом диске.

Файлы, ранее расположенные произвольно, переписываются на новый носитель плотно, без промежуточного пространства, а имеющееся свободное пространство объединяется в одну последовательную область.

Программа DSC, работающая под управлением системы, запускается двумя способами:

```
DSC  командная строка
DSC  <ВК>
```

Системнонезависимая версия программы DSC запускается по команде оператора BOO. Формат команды DSC:

```
выв.устр: метка файла /ключ=вв.устр: метка файла /ключ
```

где выв. устр: — список физических устройств, на которые передаются данные;
вв. устр: — список физических устройств, с которых копируются данные;
метка файла — идентифицирует метку тома диска или метку файла на ленте;
/ключ — один или более ключей DSC

Ключи программы *DSC*.

/AP — служит для записи файлов на ленточный том, который содержит ранее созданный DSC файл (файлы). Этот ключ используется с выводной спецификацией файла.

/BAD — применяется с выводным диском для введения информации о дефектных блоках. Этот ключ позволяет дополнять или игнорировать файл дефектных блоков, а также использовать только вручную введенную информацию о дефектных блоках. Формат:

```
BAD=MAN
    NOAUTO
    OVR
    MAN: NOAUTO
    MAN: OVR
```

где MAN позволяет вручную вводить дополнительные номера дефектных блоков в файл BADBLK.SYS;

NOAUTO игнорирует описатель дефектных блоков на диске;
OVR игнорирует описатель дефектных блоков на диске с последней дорожкой и использует последний годный к применению блок на диске, исключая последнюю дорожку;
MAN:NOAUTO вводит в файл BADBLK.SYS только вручную введенные номера дефектных блоков;
MAN:OVR позволяет вручную дополнять файл BADBLK.SYS, для которого употребляется описатель из последнего годного к использованию блока диска, исключая последнюю дорожку.

Если заданы значения MAN, MAN:NOAUTO, MAN:OVR, программа DSC отвечает подсказкой

DSC > LBN(S) =

В ответ на эту подсказку требуется ввести числа в формате N:M, где N — логический номер начального блока группы дефектных блоков, а M — число последовательных блоков в этой группе.

Если задается несколько последовательностей дефектных блоков в одной строке, то они разделяются запятой, пробелом или знаком табуляции.

/BL=N или /BL:N — позволяет установить число блоков, передаваемых за одну операцию ввода-вывода (по умолчанию — 4). Максимальное значение N для систем с ДП — 36_{10}, а для систем без ДП — 10_{10}.

/CMP — указывает на необходимость сравнения двух дисков, диска и набора лент или набора лент и диска. /CMP задается как часть выводной спецификации и используется для сравнения содержимого томов без обращения к операции копирования.

/DENS — по своему действию полностью аналогичен ключу /DEN программы BRU.

/RW — служит для перемотки каждой ленты из набора до выполнения любой операции DSC. Если /RW используется во вводной спецификации, то перематывается первая лента из набора до начала операции копирования, другие ленты перематываются по мере необходимости. Если ключ /RW определен вместе с меткой файла, то после перемотки ленты программа DSC ищет указанный файл. Если ключ /RW употребляется в выводной спецификации, то перемотка ленты происходит до операции копирования или сравнения.

/VE — служит для проверки содержимого томов после копирования. Этот ключ используется с дисками в выводной части командной строки.

Для системнонезависимой версии программы DSCSYS используются дополнительные ключи. Употребление

ключей /VEC и /CSR аналогично их применению системнонезависимой программой BAD.

Ключ /UNIT=N — изменяет номер устройства на номер, допустимый в программе DSCSYS. Системнонезависимой версии программы доступны устройства с логическими номерами 0,1. Ключом /UNIT указывается фактический номер устройства N, с которым будет работать программа DSCSYS.

Пример:

```
DSC > DP1: /UNIT=4
DSC > DP1:=DP0:
```

Данные будут копироваться с устройства DP0, на DP1, которому назначено устройство с номером 4.

6.5. ОТЛАДОЧНЫЕ ПРОГРАММЫ

В состав отладочных программ, рассматриваемых в данной главе, входят программа-отладчик (ODT) программа изменения объектного модуля (PAT) и программа изменения файлов образов задач (ZAP).

6.5.1. ПРОГРАММА-ОТЛАДЧИК ODT

Программа ODT позволяет отлаживать оттранслированные и скомпонованные программы. Работая в режиме диалога с отладочной программой, пользователь может:

напечатать содержимое любой ячейки в объектной программе для просмотра или изменения;

выполнить всю или некоторую часть объектной программы, используя точки останова;

найти слова в объектной программе по указанным двоичным образам;

найти в объектной программе инструкции, которые ссылаются на указанный адрес;

вычислить смещения адресов относительно счетчика инструкций и смещения переходов внутри объектной программы;

заполнить указанный пользователем блок слов или байтов нужными значениями;

распечатать указанный пользователем блок слов или байтов для исследования.

Во время построения задачи программа-отладчик компонуется совместно с программой пользователя (ключ /DA — гл. 5). Программа ODT является монитором задачи пользователя и выполняется с ее привилегиями.

Для упрощения дальнейшего изложения введем некоторые понятия и определения. *Точки останова* — указанные пользователем адреса, при достижении которых выполнение программы должно быть приостановлено для взаимодействия программиста с задачей, т. е. для просмотра и исправления ячеек памяти, поиска переменных и т. д. Задача после запуска выполняется до тех пор, пока не встретится точка останова, в которой управление передается отладчику, он выдает подсказку и ждет ввода команды пользователя. Программа ODT допускает одновременно до 8 точек останова.

Регистры смещения — ячейки отладчика, содержащие абсолютные базовые адреса объектных модулей после их размещения в памяти. Транслятор, создавая объектный модуль, считает равным нулю базовый адрес каждой программной секции. После компоновки модулей большинство значений внутри них изменяются на постоянную величину, которая является базовым адресом или относительным смещением объектного модуля. В процессе отладки эти смещения нужно постоянно вычитать из абсолютных адресов, чтобы установить соответствие между адресами модуля и листинга. Программа ODT автоматически вычисляет смещение каждого объектного модуля благодаря наличию 8 регистров смещения (⊙R0—⊙R7), в которые записываются относительные смещения модулей. Смещение каждого модуля берется из карты памяти MAP и заносится в соответствующий регистр смещения.

Открытая ячейка — слово или байт, содержимое которых напечатано для просмотра и может быть изменено. Для открытия ячейки нужно указать ее адрес и одну из команд открытия.

Для определения типов выражения адреса введем некоторые обозначения:

a — аргумент, используемый для указания адреса ячейки задачи;

n — восьмеричная цифра от 0 до 7;

к — восьмеричное число из 6 цифр ($к \leqslant 177777$) или выражение, которое приводится к этому числу.

Адрес открытой ячейки может определяться следующим образом: *к* — значением адреса *a* является значение *к*; *n*, *к* значением *a* является значение *к* плюс содержимое регистра смещения с номером *n* ($0 \leqslant n \leqslant 7$). В выражении

адреса также может использоваться предварительно заполненный регистр константы С.

Знаки и символы, являющиеся командами и операторами программы-отладчика, приведены ниже.

Формат	Значение
+ или пробел	Арифметический оператор. Сумма предшествующего аргумента и следующего за ним образует текущий аргумент
—	Арифметический оператор. Вычитание следующего аргумента из предыдущего образует текущий аргумент
	Оператор регистра смещения. Использует предыдущую восьмеричную цифру для ссылки к одному из восьми регистров смещения отладчика и берет содержимое этого регистра и значение аргумента, следующего за запятой, чтобы образовать текущий аргумент
*	Эта команда используется в образовании аргументов RADIX-50
. (точка)	Оператор текущей ячейки. Вызывает адрес последней явно открытой ячейки, который должен быть использован как текущий адрес для операций отладчика — адрес, предполагаемый командой < для возврата к предыдущей последовательности открытых ячеек. Этот адрес подразумевается также в использовании команд: / ' (апостроф), « (кавычки), %, <LF>
, (точка с запятой)	Разделитель аргументов. Разделяет аргументы, используется при формировании выражения адреса или значения регистра программы-отладчика
<CR> или K<CR>	Закрывает текущую открытую ячейку. В формате K<CR>, K<LF>, K↑, K←, K@, K>, K< значение K заменяет содержимое текущей открытой ячейки, прежде чем она будет закрыта
<LF> или K<LF>	Закрывает открытую в настоящее время ячейку, открывает следующую последовательную ячейку, т. е. печатает ее содержимое
↑ или K↑	Закрывает открытую ячейку и печатает содержимое предшествующей ячейки (На клавиатурах некоторых устройств вместо знака ↑ используется знак ^)
← или K←	Берет содержимое открытой ячейки как смещение относительно счетчика адреса и вычисляет адрес следующей ячейки, которая должна быть открыта; закрывает открытую текущую ячейку и печатает содержимое новой ячейки по вычисленному адресу (На клавиатурах некоторых устройств вместо знака ← используется знак —)
@ или K@	Берет содержимое открытой ячейки как абсолютный адрес, закрывает текущую ячейку и печатает содержимое ячейки по этому абсолютному адресу
> или K>	Берет младший байт открытой ячейки как смещение относительного перехода к адресу следующей ячейки, которая должна быть открыта. Вычисляет этот адрес, закрывает открытую ячейку и печатает содержимое ячейки по вычисленному адресу. Вычисление адреса происходит следующим образом: берется младший байт открытой ячейки, умножается на 2, к результату прибавляется 2 и полученная сумма прибавляется к адресу текущей открытой ячейки

< или K<	Закрывает открытую ячейку и заново открывает последнюю явно открытую ячейку, предшествующую последней из команд ←, @, <
⊙n	Адрес одного из восьми универсальных регистров программы, где n — номер регистра от 0 до 7
⊙x или ⊙nx	Адрес одного из 16 типов специальных внутренних регистров отладчика. В формате ⊙nx восьмеричная цифра n определяет номер регистра внутри группы данного типа; x является одним из следующих алфавитных указателей регистров: S — регистр состояния процессора (обозначается PS), который сохраняется отладчиком, когда появляется точка останова или ошибка программы пользователя; W — регистр слова состояния директивы ⊙DSW задачи; A — регистр аргумента поиска; M — регистр маски поиска; L — регистр нижнего предела памяти; H — регистр верхнего предела памяти; C — регистр константы; Q — регистр величины; F — регистр формата; X — векторный регистр повторной входимости; nB — регистры адреса точки останова; nG — регистры счетчика продолжения с точек останова; nI — регистры инструкций точек останова
nR	Регистры смещения
nV	Векторные регистры синхронного системного прерывания
nE	Регистры содержимого стека синхронного системного прерывания
nD	Регистры логического номера устройства
C	Оператор регистра константы. Представляет содержимое специального регистра ⊙C
Q	Оператор регистра величины. Представляет содержимое специального регистра ⊙Q
" или a"	Оператор вывода слов в символьном коде. Интерпретирует в символьном коде и печатает содержимое текущей открытой ячейки или последней открытой ячейки как два символа в символьном коде и запоминает это слово в регистре величины ⊙Q. В формате a" значение a берется как адрес ячейки, содержимое которой должно быть интерпретировано и распечатано
' или a'	Оператор вывода байтов в символьном коде. Интерпретирует и печатает содержимое текущей открытой или последней открытой ячейки как один символ в символьном коде и запоминает это слово в регистре величины ⊙Q. В формате a' значение a берется как адрес ячейки, которая должна быть интерпретирована и распечатана
% или a%	Оператор вывода слов в коде RADIX-50. Интерпретирует и печатает содержимое текущей открытой или последней открытой ячейки как три символа в коде RADIX-50 и запоминает это слово в регистре величины ⊙Q. В формате a% значение a берется как адрес ячейки, которая должна быть интерпретирована и распечатана
/	Оператор вывода восьмеричных слов. Распечатывает содержимое последнего открытого слова и запоминает это содержимое в регистре величины ⊙Q. В формате a/ значение a берется как адрес слова, которое должно быть открыто
\	Оператор вывода восьмеричных байтов. Распечатывает со-

или	держимое последнего открытого байта и запоминает это
a\	содержимое в регистре величины ⊙Q. В формате *a*\ значение *a* берется как адрес байта, который должен быть открыт
K=	Представляет выражение K в виде 6-значного восьмеричного числа, печатает его и запоминает в регистре величины
8 или 9	Отменяет текущую команду. Десятичные цифры 8 и 9 являются недопустимыми знаками. Появление 8 или 9 заставляет отладчик игнорировать текущую команду, поэтому для отмены текущей команды достаточно напечатать 8 или 9
B	Удаляет все точки останова из задачи пользователя
nB	Удаляет n-ю точку останова из задачи пользователя
a;B	Устанавливает следующую последовательную точку останова в задаче пользователя по адресу *a*
a; nB	Устанавливает n-ю точку останова по адресу *a* в задаче пользователя
K	Используя тот регистр смещения, содержимое которого меньше или равно адресу текущей открытой ячейки, вычисляет разность (смещение в байтах) между адресом текущей открытой ячейки и величиной, содержащейся в выборочном регистре смещения, печатает это смещение и запоминает его в регистре величины
nK	Вычисляет смещение (в байтах) между адресом текущей открытой или последней открытой ячейки и величиной, содержащейся в регистре смещения n, печатает это смещение и запоминает его в регистре величины
a,nK	Вычисляет смещение (в байтах) между адресом *a* и величиной, содержащейся в регистре смещения n, печатает это смещение и запоминает его в регистре величины
F или kF	Заполняет область памяти, заданной регистрами пределов памяти (⊙L и ⊙H), значением, указанным в регистре ⊙A или значением k
G или *a*G	Устанавливает инструкции BPT или восстанавливает инструкции BPT по всем ячейкам точек останова в образе задачи, восстанавливает слово состояния процессора и программные регистры пользователя, затем начинает выполнение с адреса, указанного в счетчике инструкций программы пользователя (⊙7). В формате *a*G значение *a* заменяет текущее содержание седьмого регистра, прежде чем начнется выполнение задачи
*a*0 или *a*; KO	Вычисляет и печатает смещение от адреса текущей открытой ячейки 2 до адреса *a* и 8-разрядное смещение перехода. Команда *a*;KO вычисляет и печатает смещение относительно PC и смещение относительного перехода от адреса *a* до адреса K.
KP или P	Продолжает выполнение программы пользователя от текущей точки останова и останавливает выполнение, когда встретится следующая точка останова или когда задача завершится. В формате KP продолжает выполнение программы от текущей точки останова и останавливает выполнение в точке останова при прохождении через нее в K-й раз
R	Устанавливает все регистры смещения в —1, т. е. максимальное значение адреса равно 177777_8
nR	Устанавливает в n-й регистр смещения —1, т. е. максимальное значение адреса равно 177777_8
a;R	Устанавливает следующий последовательный регистр смещения на значение адреса *a*
a;nR	Устанавливает регистр смещения n на значение адреса *a*

S или nS	Выполняет одну или n инструкций и печатает адрес следующей инструкции, которая должна быть выполнена
W или kW	В границах памяти, указанных в регистре нижнего предела памяти ⊙L и в регистре верхнего предела памяти ⊙H, отыскивает слова по указанному аргументу поиска, содержащемуся в регистре аргумента поиска ⊙A. Каждое слово памяти и аргумент поиска сравниваются по маске, указанной в регистре маски поиска ⊙M.
m;W или m;kW	В случае совпадения печатается адрес соответствующей ячейки и ее содержимое. В формате kW значение k заменяет текущее содержание ⊙A перед началом поиска. В формате m;W значение m заменяет текущее содержимое ⊙M. В формате m;kW значения k и m заменяют текущее содержимое @A и ⊙M соответственно
E или kE или m;E или m;kE	В границах памяти, указанных в регистре нижнего предела памяти ⊙L и в регистре верхнего предела памяти ⊙H, осуществляет поиск слов, которые ссылаются на исполнительный адрес, указанный в регистре аргумента поиска ⊙A, маскируемых значением, указанным в регистре маски ⊙M. Эти слова могут быть исполнительным адресом, относительным адресом либо смещением относительного перехода к исполнительному адресу, указанному в ⊙A. В формате kE значение k заменяет текущее содержимое ⊙A перед началом поиска. В формате m;E значение m заменяет текущее содержимое ⊙M перед началом поиска. В формате m;kE значения m и k заменяют текущее содержимое регистров ⊙A и ⊙M соответственно
L или KL или *a*;L или *a*;kL или n;*a*;kL	Распечатывает все ячейки в границах памяти, указанных в регистре нижнего предела памяти ⊙L и в регистре верхнего предела памяти ⊙H, используя устройство печати, указанное в регистре логического номера устройства ⊙ID. В формате kL значение k заменяет текущее содержимое ⊙H перед началом операции. В формате *a*;L значение *a* заменяет текущее содержимое ⊙L перед началом операции. В формате *a*;kL значения *a* и k заменяют текущее содержимое ⊙L и ⊙H соответственно. В формате n;*a*;kL значение n указывает регистр ⊙nD, содержащий логический номер устройства, на которое должна быть выведена распечатка
F или kF	Заполняет содержимым регистра аргумента поиска ⊙A ячейки памяти внутри пределов памяти, указанных в регистре нижнего предела памяти ⊙L и в регистре верхнего предела памяти ⊙H. В формате kF значение k заменяет текущее содержимое регистра ⊙A, перед тем как начнется операция заполнения.
V	Разрешает обращение из программы ODT ко всем векторам SST и внесение адресов входных точек ловушек ODT в таблицу, используемую директивой управляющей программы SVDB⊙
X	Выход из отладчика и возврат к управляющей программе OCPB

Примечание. При поиске под маской производится сравнение слов памяти и аргумента только в тех разрядах, которые установлены в 1 в маске, все другие разряды игнорируются. В границах памяти, указанных в регистре нижнего предела памяти ⊙L и в регистре верхнего предела памяти ⊙H, осуществляет поиск слов, которые не соответствуют аргументу поиска, указанному в регистре аргумента поиска ⊙A

6.5.2. ПРОГРАММА ИЗМЕНЕНИЯ ОБЪЕКТНОГО МОДУЛЯ PAT

Программа PAT позволяет изменять или корректировать коды в перемещаемом двоичном модуле. Программа PAT считывает файл, содержащий корректирующую информацию, и обрабатывает на основе этой информации объектный модуль. Данные корректировки готовятся в исходном виде, а затем транслируются макроассемблером.

Программа PAT вызывается одним из методов, описанных в разд. 6.1.

Формат командной строки PAT:

[вывф] = ввф [/CS [:число]], файл корректировки [/CS [:число]]

где вывф — спецификация выводного файла, если она не указана, то выводной файл не создается;

ввф — спецификация вводного файла, который может содержать один или более объектных модулей;

файл корректировки — спецификация файла корректировки, который содержит корректировку одного модуля из вводного файла;

/CS — ключ контрольной суммы, по которому создается восьмеричное значение суммы всех двоичных данных файла;

число — восьмеричное значение, в котором программа PAT сравнивает вычисленное значение контрольной суммы.

PAT выполняет корректировку относительно базы вводного модуля, используя дополнения и исправления из файла корректировки. Во время работы программы PAT обнаруженные в файле корректировки новые глобальные символы добавляются в таблицу символов модуля. Глобальные символы в файле корректировки, совпадающие с глобальными символами вводного файла, заменяют их, определяя эти символы как относительные либо как абсолютные.

6.5.3. ПРОГРАММА ИЗМЕНЕНИЯ ФАЙЛОВ ОБРАЗОВ ЗАДАЧ *ZAP*

Обслуживающая программа ZAP позволяет проверять и изменять файлы образов задач и данных на томе с файловой структурой ОСРВ. Работа программы ZAP во многом похожа на работу программы-отладчика, поэтому предварительно рекомендуем ознакомиться с п. 6.5 ZAP позволяет:

распечатывать и изменять слова и байты в образах задач на диске;

распечатывать дисковые блоки и адреса сегментов задач.

При этом программа ZAP использует ключи, односимвольные команды и специальные внутренние регистры (аналогично отладчику ODT). Программа ZAP может быть вызвана любым способом из перечисленных в п. 6.1., однако наиболее употребимый следующий:

```
> ZAP
или
ZAP> спец.ф [/ключ]
```

Ключи программы *ZAP*.

/AB — обрабатывает адреса, введенные в командной строке, как абсолютные байтовые адреса внутри файла. Этот ключ указывается обязательно, если файл не является образом задачи.

/RO — обеспечивает выполнение функций ZAP без корректировки информации на диске.

/LI — выдает начальный блок файла и граничные адреса для каждого сегмента перекрытия в следующем виде:

```
ssssss: aaaaaa — bbbbbb
```

где ssssss — указывает начальный блок в восьмеричном виде;
aaaaaa — указывает нижнюю границу адреса в восьмеричном виде;
bbbbbb — указывает верхнюю границу адреса в восьмеричном виде.

Режимы адресации в образе задачи. Как было показано в п. 6.5.1, адреса листинга ассемблера не соответствуют фактическим адресам в образе задачи, в котором большинство значений смещено на константу, являющуюся абсолютным базовым адресом программной секции после ее размещения. Иначе эта константа называется относительным смещением программной секции. Для приведения в соответствие адресов листинга с адресами в образе задачи используются регистры смещения 0R—7R.

Программа ZAP имеет два режима адресации к образу задачи: абсолютную и адресацию относительно образа задачи.

При абсолютной адресации (с ключом /AB) программа ZAP интерпретирует первый адрес в файле как сегмент 1 с начальным адресом 000000, который используется как базовый при интерпретации других вводимых адресов. При этом режиме можно обращаться ко всем байтам файла, в частности, к блокам заголовка образа задачи. Этот способ обычно применяется для односегментных задач.

В способе адресации относительно образа задачи программа ZAP позволяет адресоваться к ячейкам, используя номер блока и относительные смещения, кото-

рые берутся из карты распределения памяти построителя задач.

Команды программы *ZAP*. Существует три группы команд ZAP: команды открытия-закрытия ячеек; основные команды и возврат каретки.

Командой открытия-закрытия ячеек является один из следующих символов: /, ", %, \, ', ↑ или ^, ← или —, @, >, <. Эти команды полностью соответствуют одноименным командам ODT (см. п. 6.5.1).

Программа ZAP имеет шесть однозначных основных команд (X, K, O, =, V, R). Команды X, K, O, =, R работают так же, как в программе ODT, команда V проверяет содержимое текущей ячейки.

Возврат каретки <CR> заставляет закрыть текущую измененную ячейку. Два последовательно введенных символа <CR> открывают следующую ячейку в файле. В отличие от ODT команды ZAP действуют только после символа <CR>.

Программа ZAP использует специальным образом несколько фиксированных ячеек памяти, называемых внутренними регистрами. Имеется 8 регистров смещения (0R—7R), постоянный регистр или регистр константы C, регистр формата F, регистр величины Q. Использование этих регистров, арифметических операторов (+, —, *) и символов-разделителей командной строки (; ,) полностью совпадает с их использованием программой ODT.

6.6. ПРОГРАММА-БИБЛИОТЕКАРЬ (*LBR*)

Обслуживающая программа-библиотекарь позволяет создавать, корректировать, модифицировать и распечатывать библиотечные файлы — наборы данных прямого доступа, содержащие файлы одного типа.

Библиотечные файлы содержат две справочные таблицы: таблицу входных точек (EPT) и таблицу имен модулей (MNT). Таблица EPT включает имена точек входа, которые в исходных программах определены как глобальные символы, а таблица MNT — имена модулей в библиотеке.

Существуют три типа библиотек: объектная (.OLB), макро (.MLB) и универсальная (.ULB).

Объектные библиотеки состоят из объектных модулей. Программа-библиотекарь обращается к модулям по имени, а ссылки к ним из исходных программ осуществля-

ются по именам точек входа. Объектные библиотеки используются построителем задач как входные данные.

Макробиблиотеки включают наборы макрокоманд в исходном виде. Имена модулей берутся из директив MACRO [4]. Макробиблиотеки используются макроассемблером в качестве входных данных.

Универсальные библиотеки содержат модули, состоящие из любых файлов одинакового типа программ или текстов. Из программы к таким модулям можно ссылаться, используя подпрограмму ⊙ULA, находящуюся в системной библиотеке (SYSLIB.OLB).

Программа LBR может вызываться любым способом, описанным ранее (п. 6.1). Формат командной строки программы-библиотекаря:

вывф[, файл листинга] = ввф1[, ввф2, ... , ввфN]

Ключи программы *LBR*

/CO — создает новую версию библиотечного файла, физически удаляя все логически вычеркнутые записи, сдвигая свободное пространство в конец файла и делая его пригодным для включения новых модулей. С помощью десятичных числовых ключей можно изменить характеристики файла. Формат:

вывф/CO[:РБФ] [:РТВХ] [:РТИМ] = ввф

где РБФ — размер нового библиотечного файла в 256-словных блоках. По умолчанию размер старого файла;

РТВХ — число записей, которые можно разместить в таблице входных точек (не более 4096). По умолчанию берется значение из старой библиотеки;

РТИМ — число записей в таблице имен модулей (не более 4096). По умолчанию берется значение из старой библиотеки. Числа РТВХ и РТИМ берутся кратными 64.

/CR — создает непрерывный библиотечный файл на диске, в том числе заголовок файла и соответствующие таблицы. Формат:

вывф/CR :РБФ :РТВХ :РТИМ :ТИПБ :ТИПВФ

где параметры РБФ, РТВХ, РТИМ имеют то же значение, что и у ключа /CO. По умолчанию параметр РБФ равен 100; РТВХ —512, РТИМ —256, (OBJ — для объектных библиотек, MAC — для макробиблиотек и UNI для универсальных);

ТИПБ — тип создаваемой библиотеки;

ТИПВФ — умолчание типа файла для создаваемой

универсальной библиотеки, если параметр не задан, то по умолчанию. UNI.

/DE — производит логическое удаление библиотечных модулей и их входных точек. При логическом удалении модуль становится недоступен, однако занимает место в библиотеке. Формат:

вывф/DE :МОД1 :МОД2 ... :МОД15

где МОД — имя удаляемого модуля.

/DF — определяет умолчание типа библиотечного файла. Допустимые значения такие, как в параметре ТИПБ ключа /CR. Формат:

вывф/DF :ТИП или /DF :ТИП

/DG служит для удаления входных точек из таблицы EPT. Причем этот ключ не удаляет глобальный символ из модуля, который содержит его фактическое определение. Формат:

вывф/DG :ГЛОБЛ1 :ГЛОБЛ2 ... :ГЛОБЛ15

/EP — предоставляет возможность включить или исключить запись глобальных символов в таблицу входных точек библиотеки. Если указывается положительная форма ключа /EP, то входные точки включаются в таблицу EPT, если отрицательная /—EP или /NOEP, то не включаются. Формат:

вывф1 [/EP] = ввф [/EP] ... ввф [/EP]

Если /EP стоит в спецификации выводного файла, то программа-библиотекарь предполагает, что каждый вводной файл содержит модули с включаемыми или исключаемыми входными точками. Если /EP указан в спецификации вводного файла, то для модулей только этого файла определяются, включаются или исключаются их входные точки. Ключ /EP служит умолчанием для программы-библиотекаря.

/EX — позволяет извлекать из библиотечного файла один или несколько модулей и писать их в указанный выводной файл. Библиотечный файл остается без изменений. Для объектной и макробиблиотеки может быть указано до 8 модулей, для универсальной — только один. Формат:

вывф = ввф/EX [:МОД1 ... :МОДN]

/IN — предоставляет возможность вставлять модули в библиотечный файл. Для макробиблиотек из вводных файлов извлекаются только макроопределения верхнего уровня. Формат:

вывф [/IN] = ввф1 [ввф2 ... ввфN]

Для универсальных библиотек ключ /IN указывается в спецификации вводного файла. После ключа можно указать имя модуля и описательную информацию, которая сохраняется в заголовке модуля. Формат:

```
вывф=ввф/IN[:МОД:ОП:ОП:ОП:ОП]
```

/LI, /LE, /FU— позволяет распечатывать оглавление библиотечного файла:

/LI — распечатывает имена всех модулей библиотеки;

/LE — распечатывает имена всех модулей библиотеки и их входных точек;

/FU — распечатывает имена всех модулей библиотеки и дает полное описание каждого модуля (размер, дату включения и т. д.). Формат:

```
вывф[,файл листинга]/ключи
```

/MH — относится только к универсальным библиотекам и позволяет изменять задаваемые пользователем заголовки в описателе модуля. Формат:

```
вывф/МН:МОД:ОП:ОП:ОП:ОП
```

/RP — заменяет модули в библиотечном файле на вводные модули с теми же именами. Для макробиблиотек извлекаются макроопределения только первого уровня. Ранее существующий модуль удаляется логически, а из таблицы EPT удаляются входные точки. Ключ замены может быть указан в локальном или глобальном формате. В глобальном формате ключ /RP присоединяется к спецификациям выводного файла и предполагается, что все вводные файлы содержат модули для замены. В локальном формате ключ /RP присоединяется к спецификации вводного файла и считается, что только файлы с ключом /RP содержат модули для замены. Формат:

```
вывф[/RP]=ввф1[/RP][,ввф2[/RP] ... ввфN[/RP]]
```

Используя ключ /RP для универсальной библиотеки, можно, как и для ключа /IN, в командной строке указать имя модуля и описательную информацию.

/SP — функционирует так же, как и в программе PIP (см. п. 6.2.1).

/SS — используется при операциях вставки для установки бита выборочного поиска в байте атрибутов заголовков объектных модулей при включении их в объектную библиотеку.

Объектные модули с атрибутом выборочного поиска обрабатываются построителем задач специальным образом. Глобальные символы, определяемые в таких модулях, включаются построителем задач в таблицу символов (STB) в том случае, если на них есть ссылки из других задач. Формат:

```
вывф=ввф1/SS[ввф2[/SS],...,ввфN[/SS]]
```

/SZ — позволяет уменьшить размеры макроопределений, удаляя из них пробелы, знаки табуляции, пустые строки и комментарии. Ключ сжатия употребляется в локальном и глобальном формате (отличия форматов такие же, как и в ключе /RP). Формат:

```
вывф[/SZ]=ввф1[/SZ][,ввф2[/SZ],...,ввфN[/SZ]]
```

7.

ГЕНЕРАЦИЯ ОСРВ

Процедура *генерации ОСРВ* представляет собой последовательность действий, направленных на создание конкретного варианта целевой системы. По своим основным действиям процедура генерации разбита на три этапа, называемые фазами генерации:

ф а з а 1 — трансляция модулей управляющей программы, в процессе которой происходит настройка в соответствии с выбранными возможностями;

ф а з а 2 — построение управляющей программы и системных привилегированных задач, необходимых для функционирования системы; создание с помощью программы VMR файла образа системы, готового к выполнению;

ф а з а 3 — построение обслуживающих непривилегированных задач.

Генерация может выполняться сразу или раздельно по фазам. Если для фазы 2 всегда необходимы результаты трансляции фазы 1, то фаза 3 является в этом смысле независимой как от фазы 1, так и от фазы 2, и может быть выполнена отдельно в любое время.

В результате генерации ОСРВ можно построить системы, отличающиеся друг от друга возможностями управляющей программы, набором внешних устройств, распределением оперативной памяти (ОП) и т. п.

С точки зрения программиста ОСРВ представляет собой большой комплекс программ, написанных на языке макроассемблера. Эти программы широко используют возможности языка MACRO, в первую очередь, директивы условной трансляции. В зависимости от того, определена или нет та или иная переменная условной трансляции (или от значения соответствующей переменной), макроассемблер включает в программу соответствующий код.

Переменные условной трансляции определяют параметры системы: имеется ли диспетчер памяти в процессоре, поддерживается ли выгружаемость задач, обслуживаются ли AST-прерывания, возможности терминального драйвера и т. д. Всего имеется около 100 настраивающих систему параметров. Это позволяет генерировать чрезвычайно широкий спектр систем, включая в каждую из них только необходимые функции. При такой организации важно, чтобы выбранные значения переменных не противоречили друг другу и чтобы все модули управляющей программы системы транслировались с одним и тем же набором параметров. Для этого, а также для облегчения процедуры генерации в ОСРВ предусмотрены косвенные командные файлы генерации системы. Командные файлы генерации используют возможности процессора косвенного командного файла.

Запустив косвенный командный файл, пользователь лишь отвечает на вопросы о необходимости тех или иных характеристик системы, наличии и адресах регистров и векторов тех или иных внешних устройств и т. д. В результате командный файл генерации, контролируя непротиворечивость ответов, создает файл параметров, определяющих характеристики генерируемой системы (ОСМС.МАС).

Одновременно создаются командные файлы для трансляции и построения системы. Первый из этих файлов обеспечивает трансляцию каждого необходимого модуля управляющей программы системы с одним и тем же файлом параметров (фаза 1). Второй — файл построения — создает на диске файл-образ управляющей программы системы OCPBCM.TSK и файл значений ее глобальных имен OCPBCM.STB (фаза 2).

7.1. ПАРАМЕТРЫ ГЕНЕРАЦИИ СИСТЕМЫ (ФАЗА 1)

Рассмотрим подробнее параметры, устанавливаемые при генерации системы. Имеются следующие группы параметров:

конфигурации оборудования;

определяющие допустимые запросы к управляющей программе (директивы);

системных возможностей;

настройки ядра системы;

повышающие надежность работы системы;

размещения системы;

драйвера терминала;

возможности развития системы (подключение дополнительных устройств);

определяющие дополнительные операции на устройствах.

Параметры конфигурации оборудования. В процессе диалога командный файл генерации запрашивает тип процессора (СМ2103, СМ2104, СМ2420, СМ1300 и т. д.), в результате определяются следующие характеристики: наличие регистра переключателей в процессоре, наличие расширителя арифметики с фиксированной запятой (EAE) и аппаратуры для работы с числами с плавающей запятой (FIS или FPP).

Отдельно запрашиваются тип таймера (программируемый или сетевой), частота сети, размер ОП, наличие аппаратуры контроля паритета памяти, наибольший адрес вектора прерывания, используемый внешними устройствами, типы и число контроллеров устройств, используемых в системе, двухпроцессорная работа ОСРВ.

Размер ОП, с которой будет работать ОСРВ, зависит от выбираемого типа процессора. Для СМ2103 и СМ1300, не имеющих диспетчер памяти, система ОСРВ обеспечивает работу с ОП в пределах 56 Кбайт, для СМ2104 и СМ2420, имеющих диспетчер памяти,— в пределах 248 Кбайт, для СМ2420 при наличии дополнительного оборудования (контроллера памяти) — до 4 Мбайт. Если диспетчер памяти СМ2420 осуществляет программный запрос к адресному пространству до 4 Мбайт, то контроллер памяти обеспечивает внепроцессорные запросы устройств прямого доступа к тому же объему памяти.

Выбор двухпроцессорной работы ОСРВ предполагает наличие специального двухпроцессорного комплекса с двухвходовой ОП: на первом (основном) процессоре обрабатываются прерывания от операций ввода-вывода и выполняется самая приоритетная задача, а на втором — выполняются менее приоритетные задачи. Такое перераспределение задач обеспечивает более высокую производительность комплекса.

При выборе внешних устройств запрашивается перечень мнемонических имен внешних устройств, с которыми будет работать система. Для каждого выбранного типа устройства указывается число контроллеров.

В номенклатуре СМ ЭВМ имеется несколько устройств одного типа.

Ниже приведены наиболее распространенные из устройств, с которыми может работать ОСРВ.

CR — устройство ввода с перфокарт типа СМ6102
CT — кассетная магнитная лента типа СМ5208
DF — магнитный диск с фиксированными головками типа СМ5500
DK — кассетный магнитный диск (2,4 Мбайт) типа СМ5402
DM — кассетный магнитный диск (14 Мбайт) типа СМ5408
DP — пакетный магнитный диск (20 Мбайт) типа СМ5407
DR — пакетный магнитный диск (80 Мбайт) типа СМ5412
DX — гибкий магнитный диск типа СМ5603
DY — гибкий магнитный диск с двойной плотностью записи;
GR — устройство отображения графической информации типа СМ7300
LP — алфавитно-цифровое печатающее устройство типа СМ6305
MT — магнитная лента с плотностью записи 32 бит/мм типа СМ5301
MM — магнитная лента с плотностью записи 64 бит/мм
PP — устройство вывода на перфоленту типа СМ6204
PR — устройство ввода с перфоленты типа СМ6204
XL — адаптер дистанционной связи СМ8502
YL — терминал, подключенный через адаптер ИРПР или ИРПС
YH — асинхронный мультиплексор МПД-А типа СМ8514 на 16 линий
YZ — асинхронный мультиплексор на 8 линий
XW — синхронный адаптер дистанционной связи АДС-С
KR (KW) — дуплексный регистр типа К5-СМ
BS — переключатель общей шины СМ4501
BW — виртуальное окно на общей шине, образуемое через АМС
CW — физическое окно на общей шине, образуемое через АМС
CO — псевдоустройство — консоль оператора
NL — пустое псевдоустройство

Параметры, определяющие допустимые запросы к управляющей программе (директивы). Если генерируется система общего назначения и нет особых требований к размеру оперативной памяти, то обычно в управляющую программу включают все директивы. Однако пользователь при генерации может управлять набором используемых директив. Система с диспетчером памяти может, например, иметь или не иметь группу директив отображения виртуального адресного пространства — создать, присоединить, отсоединить район, отобразить, отсоединить, создать адресное окно. Отдельно можно управлять включением директив SREF⊙ и GMCX⊙.

Независимо от типа системы можно управлять включением следующих директив:

получить параметры раздела;
получить параметры задачи;
послать/получить данные;
изменить приоритет задачи;
расширить память задачи;
присоединить вектор прерывания;
получить значение консольных переключателей;
установить системное время;
директива синхронизации задач по биту STOP;
асинхронное прерывание по аварийному завершению задачи;

директивы, порождающие отношения «родитель — потомок»;

выборочное удаление запросов на объявление временных интервалов директивы MRKT⊙ (CMKT⊙).

При этом нужно помнить, что в системе с неполным набором директив нельзя выполнять задачи, использующие отсутствующие директивы. Так, системные обслуживающие задачи и программа связи с оператором MCR используют директивы: «получить параметры раздела» и «получить параметры задачи».

Командный файл генерации контролирует непротиворечивость выбора параметров системы путем выдачи предупреждающих сообщений и автоматического включения необходимых возможностей (в частности, директив). Например, если пользователь выбирает возможность работы с ПП СТО/РВ [2], то в систему автоматически включаются директивы: отменить запросы директивы MRKT⊙, получить параметры раздела, получить параметры задачи, расширить память задачи. Следовательно, при генерации в систему автоматически включаются те возможности, которые необходимы для правильной работы выбираемых компонентов системы, даже если ранее от этих возможностей отказались.

Параметры системных возможностей. Большинство из описанных ниже возможностей обычно включается в систему общего назначения, а в целевых проблемных системах некоторые из них могут исключаться.

1. Файловые возможности. Выбирается поддержка файловой структуры ОСРВ на диске и магнитной ленте, требуемая версия файлового процессора (малая, средняя, большая), свойства системы управления файлами (с резидентной в памяти библиотекой или нет), уровень буферизации для магнитных лент, блокировка записей для СУД.

2. Диспетчер очередей (QMG). Диспетчер очередей включается в большинство систем для обслуживания параллельных запросов вывода на АЦПУ с помощью механизма очередей. Если система не содержит диспетчера очередей, то вместо него может использоваться задача последовательной системной распечатки (PRT...).

3. Задача аварийной распечатки памяти (PMD). Позволяет выводить на печать указанные в задачах области памяти и записывать в файл содержание памяти при аварийных завершениях задач.

4. Многопользовательская защита. Каждый пользователь защищен от воздействия операций других пользователей. Это обеспечивается как средствами самой управ-

ляющей программы, так и наличием в этом случае дополнительных системных привилегированных и непривилегированных задач.

5. Системно-управляемые разделы. В них возможно динамическое распределение памяти для задач, как это описано в гл. 1. Задача, выполняющаяся в системно-управляемом разделе, может увеличивать или уменьшать свое виртуальное адресное пространство, динамически размещать и передавать друг другу блоки памяти (районы) и т. д. Один большой системно-управляемый раздел удобен в системе с несколькими терминалами, используемой для разработки программ. Однако время загрузки задачи в системно-управляемый раздел может зависеть от состояния других задач и может возрастать до величин, неприемлемых в конкретной системе реального времени.

6. Возможность установки — запуска — завершения задачи (RUN⊙) и логическое назначение устройств. Возможность установки — запуска — завершения задачи позволяет экономить динамическую память, а логическое назначение устройств дает возможность обращаться к физическим устройствам в системе по их логическим именам.

Параметры настройки ядра системы. Сюда входят параметры, определяющие возможность работы задачи уплотнения памяти (SHF), число предварительно размещенных в очереди запросов ввода-вывода, интервал времени и границы приоритетов задач для круговой диспетчеризации, размер динамической памяти и т. д. От этих параметров зависит как время реакции самой системы, так и время выполнения пользовательских и системных задач. Так, предварительно размещенные в памяти запросы ввода-вывода влияют на скорость обслуживания этих запросов. Задача уплотнения памяти устраняет фрагментацию в памяти, увеличивая тем самым количество загруженных задач. Параметры круговой диспетчеризации и приоритет обмена с диском влияют на прохождение задач в системе.

Размер динамической памяти имеет важное значение для нормального функционирования системы: нехватка памяти может приводить к замедлению обслуживания запросов и «зависанию» системы.

Параметры, повышающие надежность работы системы. К этим параметрам относятся параметры, определяющие возможность контроля адресов задачи, вывода сообщений об аварийном завершении задач (TKTN), вывода сообщений о неготовности устройств, контроля заверше-

ния операций ввода-вывода, восстановления по отказу питания, регистрации ошибок оборудования, возможность анализа содержимого памяти после аварии системы, возможность резервирования однотипных дисковых устройств.

С помощью контроля адресов определяется принадлежность адресов адресному пространству задачи. Задача TKTN дает возможность пользователю получать информацию об аварийном завершении задач. Благодаря выводу сообщений о неготовности устройств оператор может вовремя вмешаться в работу внешних устройств с тем, чтобы продолжить выполнение задач, работающих с внешними устройствами. Контроль завершения операций ввода-вывода обеспечивает слежение в системе за правильностью выполнения операций ввода-вывода на устройствах при завершении задачи.

Восстановление по отказу питания позволяет на вычислительных комплексах с неразрушающимся по отказу питания содержимым ОП сохранить состояние операционной системы в момент отказа питания и продолжить работу системы после его восстановления.

Благодаря регистрации операционной системой ошибок оборудования технический обслуживающий персонал может вовремя проводить профилактическую работу по устранению неисправностей.

Средства сохранения памяти аварийного состояния системы и задача анализа аварийного состояния (CDA) необходимы в случае аварийных ситуаций в системе, особенно при отладке системных модулей, привилегированных задач и драйверов. Они позволяют сохранить в момент аварии содержимое системных таблиц, состояние регистров, стека и другие параметры на внешнем устройстве (магнитной ленте или диске) для последующего анализа.

Выбор резервирования магнитных дисков повышает надежность работы системы за счет дублированного хранения информации и возможности динамической реконфигурации используемых дисковых устройств без завершения выполняющихся задач.

Параметры размещения системы. Эти параметры определяют размещение компонентов ядра системы. Такие важнейшие компоненты системы, как загрузчик задач, модули, реализующие директивы управляющей программы, драйверы внешних устройств могут находиться либо в ядре системы, либо быть вынесены за его пределы в соответствующие им разделы ОП, уменьшая тем самым размер

ядра системы. От параметров, определяющих размещение перечисленных компонентов, зависит возможный размер динамической памяти, допустимый размер привилегированных задач. Меньшее ядро системы позволяет увеличить размер динамической памяти.

Параметром, влияющим на размер динамической памяти и допустимый размер привилегированных задач, является максимально допустимый размер адресного пространства управляющей программы системы. Чем больше этот размер, тем больше памяти может быть отведено под динамическую область, однако при этом допустимый размер системных привилегированных задач уменьшается. Это связано с распределением регистров диспетчера памяти, которые используются для адресации управляющей программы и привилегированных задач.

Параметры драйвера терминала. Эти параметры определяют режим работы (дуплексный или полудуплексный) и различные характеристики терминала, возможности изменения этих характеристик и распознавания ESC-последовательностей.

Параметры возможностей развития системы. К этим параметрам относятся: возможность включения в систему драйверов внешних нестандартных устройств, включение в ядро системы подпрограмм передачи данных в словах (⊙GTWRD и ⊙PTWRD), а также возможность работы с системным отладчиком.

Пользователь может разработать и включить в операционную систему драйвер своего, нестандартного внешнего устройства. Драйвер может быть резидентным с резидентными таблицами (непосредственно включенный в ядро системы) либо загружаемым (вне ядра системы), но с резидентными таблицами и, наконец, может быть вариант загружаемого драйвера с загружаемыми таблицами, когда таблицы загружаются одновременно с драйвером.

Для пользовательских драйверов, использующих пословную передачу данных, могут быть полезны подпрограммы ⊙GTWRD и ⊙PTWRD, которые в этом случае включаются в ядро системы.

Развитие системы — введение новых драйверов в систему, отладка пользовательских привилегированных задач — требует включения в ядро системы системного отладчика (XDT).

Системный отладчик по своим возможностям аналогичен программе ODT (см. п. 6.5.1), однако он может работать с управляющей программой, драйверами и привилегированными задачами.

Параметры, определяющие дополнительные операции на устройствах. К этим параметрам относятся диагностические функции QIO (см. гл. 2) и операции контроля записи на дисках.

Диагностические функции QIO-запроса, будучи включенными в систему, расширяют число кодов функций для ряда внешних устройств (DK, DP, MT и т. д.) и позволяют выполнять специальные диагностические операции на устройствах. На использовании этих специальных функций базируется работа ППП «Прораб» и «Эксперт» [5], тестирующих внешние устройства, и задачи FMT, выполняющей форматирование дисков.

Контроль записи на диск обеспечивает надежную работу с дисковыми устройствами, дублируя каждую операцию записи на диск операцией чтения со сравнением данных в контроллере диска.

7.2. ДЕЙСТВИЯ В ФАЗЕ 2 ГЕНЕРАЦИИ ОСРВ

В фазе 2 создаются ядро целевой системы и привилегированные задачи. Основными действиями в этой фазе являются построение из библиотеки OCPBCM.OLB файла образа системы OCPBCM.SYS, соответствующего ему файла таблицы глобальных имен системы OCPBCM.STB и построение и привязка системных привилегированных задач к образу системы. Приведем перечень основных привилегированных задач, которые могут быть построены, их функции в системе:

FOCACP — процессор управления файлами на диске;
MCR — программа связи с оператором; обеспечивает обработку и выполнение команд оператора;
ICP — процессор косвенного командного файла;
TKTN — задача вывода системных сообщений;
INS — задача установки задач в системе; является внешней функцией MCR;
SAV — задача сохранения системы; является внешней функцией MCR,
MTACP — файловый процессор магнитной ленты; обеспечивает работу с файловой структурой на магнитной ленте;
PMT — задача, контролирующая динамическую память системы;
ACS — задача выделения динамического файла выгрузки задач; является внешней функцией MCR;
MOU — задача монтирования тома; является внешней функцией MCR,
BOO — задача загрузки системы; является внешней функцией MCR,
SHF — задача уплотнения памяти; перемещает задачи в системно-управляемом разделе с целью уменьшения фрагментации;
PMD — задача вывода на печать области памяти при аварийном завершении задач;
QMG — диспетчер очередей;

LOA — загрузчик загружаемых драйверов; является внешней функцией MCR;
DMO — задача демонтирования томов; является внешней функцией MCR,
UNL — задача выгрузки загружаемых драйверов, является внешней функцией MCR;
DCL — задача-интерпретатор диалогового командного языка; обеспечивает обработку и выполнение команд оператора;
INI — задача инициализации диска; является внешней функцией MCR;
ERRLOG — задача регистрации ошибок оборудования;
BRO — задача вывода сообщений на терминалы; является внешней функцией MCR;
BYE — задача выхода пользователя из системы; является внешней функцией MCR;
HELLO — задача регистрации пользователя в системе; является внешней функцией MCR;
RSV, UPD — задачи, обеспечивающие резервирование дисков;
UFD — задача, создающая каталог; является внешней функцией MCR;
SHUTUP — задача завершения работ в системе.

Для завершения генерации построенный образ системы необходимо обработать специальной программой, называемой виртуальной программой связи с оператором (VMR). Программа VMR выполняет следующие действия:

устанавливает размер динамической памяти;

распределяет ОП, разбивая ее на разделы;

связывает с системой загрузчик задач;

загружает драйверы внешних устройств;

устанавливает общие области управляющей программы, разделяемые библиотеки и системные привилегированные задачи;

фиксирует задачи (делают их резидентными в памяти);

делает задачи запускаемыми через некоторый интервал времени (команда RUN);

устанавливает функции терминального драйвера и т. д.

Возможности программы VMR во многом соответствуют возможностям программы MCR. Синтаксически команды этих программ совпадают, однако, если команды MCR воздействуют на выполняющуюся систему в оперативной памяти, то команды VMR воздействуют на образ системы, размещенный на диске.

Командный файл SYSVMR.CMD, создаваемый в фазе 2, облегчает процедуру обработки образа системы программой VMR.

При наличии в параметрах генерации отладчика управление при запуске системы передается первоначально системному отладчику, а затем осуществляется инициализация и запуск системы.

Последние действия в работающей системе выполняет задача сохранения образа системы SAVE. Цель ее — сделать систему загружаемой с помощью аппаратного загрузчика, организовать при каждом последующем запуске системы проверку наличия в конфигурации машины включенных в систему устройств, наличия на системном диске всех установленных в системе задач, запустить стандартный косвенный командный файл STARTUP.CMD. На основе этого файла можно создавать с помощью команд программы MCR и директив процессора косвенного командного файла системы различного целевого назначения, используя один и тот же файл образа системы.

В процессе эксплуатации системы пользователь (системный программист) может при необходимости провести реконфигурацию системы в ее образе на диске с помощью программы VMR или в выполняющейся системе в ОЗУ с помощью программы MCR и с последующим сохранением системы на диске командой оператора SAVE. Это делают, например, для ввода в систему дополнительного драйвера после подключения к машине дополнительного устройства.

7.3. ДЕЙСТВИЯ В ФАЗЕ 3 ГЕНЕРАЦИИ ОСРВ

Фаза 3 генерации предназначена для построения системных обслуживающих задач. Она может быть использована для построения и системных привилегированных задач.

Приведем перечень системных обслуживающих задач которые могут быть созданы в фазе 3 генерации:

BAD — задача поиска дефектных блоков на диске;
BRU — задача копирования и восстановления томов;
CDA — задача сохранения состояния системы при аварии;
CMP — задача сравнения файлов;
DSC — задача сохранения и уплотнения тома,
CRF — задача создания файла перекрестных ссылок;
DMP — задача вывода блоков файла;
EDI — строковый редактор,
FMT — задача форматирования дисков;
FLX — задача преобразования файлов;
LBR — библиотекарь,
MAC — транслятор с языка MACRO,
PAT — задача корректировки объектных модулей,
PIP — задача работы с файлами;
PRT — задача системного вывода;
SLP — пакетный редактор,

RETEX — задача редактирования и форматирования текстовой документации;
TKB — построитель задач,
VFY — задача проверки томов и файловой структуры,
VMR — виртуальная программа связи с оператором;
ZAP — задача корректировки образа задачи;
EXPERT — ППП „Эксперт"
IOX — программа проверки функционирования оборудования „Прораб",
TECO — программируемый редактор;
PRE — задача сохранения тома;
DOC — задача форматирования текстов.

7.4. СИСТЕМА ВИРТУАЛЬНОГО ДОСТУПА (СВД)

Система СВД обеспечивает возможность задачам, работающим в одной (локальной) ЭВМ распределенного многомашинного комплекса, обращаться к данным и устройствам другой (удаленной) ЭВМ. СВД позволяет также оператору с терминала одной ЭВМ использовать вычислительные ресурсы удаленной ЭВМ.

С помощью СВД доступ к устройствам удаленной машины обеспечивается посредством системной директивы QIO, поэтому задачи в ОСРВ могут обращаться к устройствам удаленной машины так же, как к устройствам ЭВМ, на которой они выполняются. Для этой цели создаются виртуальные устройства, соответствующие физическим устройствам удаленной ЭВМ.

Обращения задач к виртуальному устройству воспринимает задача-передатчик, которая через линию связи сообщается с задачей-приемником в удаленной ЭВМ. Задача-приемник выполняет операцию ввода-вывода на устройстве и возвращает результат обратно.

Работа за терминалом в локальной ЭВМ практически не отличается от работы в системе ОСРВ без СВД. Однако командные строки, введенные с терминала, обрабатываются в локальной ЭВМ специальным интерпретатором командных строк RCL, входящим в состав системы СВД. В зависимости от выданных команд интерпретатор RCL передает командные строки либо программе MCR, либо интерпретатору командных строк виртуального терминала в удаленной ЭВМ. В результате оператор работает как со своей, так и с удаленной ЭВМ.

ЭВМ многомашинного комплекса, на котором работает СВД, могут быть связаны в различные конфигурации типа «звезда», «кольцо», СВД предоставляет возможность доступа к данным и задачам любой ЭВМ с любого терминала

многомашинного комплекса, для чего необходимо создать виртуальные устройства, соответствующие маршруту передачи.

Работа оператора с системой виртуального доступа включает:

создание виртуального устройства;

связь виртуального устройства с физическим;

работу с виртуальным устройством;

уничтожение связи виртуального устройства с физическим.

Виртуальное устройство создается командой LNK CREATE. Она создает одно или несколько виртуальных устройств одного и того же типа (диск, терминал и др.). Тип устройства определяется по типу физического устройства в команде связи с физическим устройством. Формат команды:

```
LNK CRE[ATE] DD [,N1] [,N2]
```

где DD — двухбуквенное имя виртуального устройства,
N1, N2 — минимальный и максимальный номера устройств (по умолчанию N1 = 0, N2 = N1).

Например, команда LNK CREATE VD, 2, 4 создает три устройства VD2, VD3 и VD4.

В качестве устройства DDN может указываться виртуальное устройство, уже связанное с физическим устройством. Это позволяет обратиться через СВД не в соседнюю, а в следующую ЭВМ и т. д.

Связь виртуального устройства с физическим. Отображение виртуального устройства в физическое устройство удаленной ЭВМ выполняется командой LNK TIE. Каждое из виртуальных устройств, созданных командой LNK CREATE, может назначаться физическим устройствам любой из удаленных ЭВМ, для этого требуется лишь совпадение типов (допустимых функций) устройств. Первая команда LNK TIE сохраняет в структурах данных виртуального устройства характеристики физического устройства. Последующие команды LNK TIE к другим виртуальным устройствам группы контролируют соответствие типов. Например, если созданы виртуальные устройства VD0 и VD1 и устройство VD0 назначено диску DK3 соседнего узла NOD1, то устройство VD1 может быть назначено только диску. Это может быть диск того же узла NOD1 или любого другого. Не обязательно VD1 назначать диску типа DK; устройство VD1 может быть назначено, например, диску типа DP. Формат команды:

LNK TIE VUNN —NODE_DDNN
или
LNK TIE VUNN —NODE DDNN
где VUNN — имя и номер виртуального устройства,
NODE— имя узла с удаленной ЭВМ,
DDNN— имя и номер физического устройства.

Работа с виртуальным устройством. Виртуальному устройству, так же как и физическому, назначается логический номер LUN (в момент построения задачи, при её установке или по директиве ALUN ⊙ в процессе работы), и затем выполняются операции ввода-вывода с помощью директивы QIO. Система виртуального доступа, обслуживая запросы QIO от задач в локальной ЭВМ, выполняет соответствующие операции на устройствах удаленной ЭВМ и возвращает результат операции задаче. Для работы с устройством через систему виртуального доступа не требуется никакой доработки задач. Системные обслуживающие задачи, использующие логический уровень ввода-вывода, могут без изменений работать с устройствами удаленных ЭВМ.

Если виртуальному устройству назначен диск, то с этого устройства можно установить и запустить задачу. СВД обеспечивает выполнение запросов на загрузку перекрытий задач, запущенных с устройства удаленной ЭВМ. Оператор может назначать виртуальному устройству логические имена устройств (по команде ASN), переадресовывать виртуальные устройства (по команде RED), использовать другие команды оператора ОСРВ. Таким образом, СВД может использовать все программные средства ОСРВ.

Однако обслуживание запроса ввода-вывода на удаленном устройстве требует передачи массивов данных по линиям межмашинной связи, что увеличивает время выполнения запроса. В одних случаях это практически незаметно для оператора, в других — временные характеристики работы задач могут существенно изменяться. В зависимости от типа устройства могут возникать и другие особенности использования удаленных устройств.

Уничтожение связи виртуального устройства с физическим. Команда уничтожения связи виртуального устройства некоторой установки с физическим устройством удаленной ЭВМ выполняет действия, обратные действиям команды LNK TIE. Связь устройств разрывается. Однако структуры данных виртуального устройства не уничтожаются. Обращения по запросам QIO к отключенным устройствам отвергаются с выдачей в слове состояния

директивы (DSW) кода завершения IE.OFF Тип устройства не изменяется В дальнейшем виртуальное устройство может быть назначено другому физическому устройству командой LNK TIE. Формат команды·

LNK UN [TIE] VUNN

где VUNN имя и номер виртуальноro устройства

Команды LNK TIE и LNK UNTIE не отслеживают задачи с логическими номерами LUN, назначенными на виртуальное устройство. В целях уменьшения числа ошибок команда LNK UNTIE не выполняется, если виртуальное устройство присоединено или смонтировано. В случае, если виртуальное устройство переадресовано по команде оператора RED на другое устройство, оператору выдается предупреждающее сообщение.

Терминал оператора СВД может работать как в локальной, так и в удаленной ЭВМ. Подключение к удаленной ЭВМ и отключение от нее осуществляется командами RCL (подключиться к удаленной ЭВМ) и LCL (вернуться для работы в локальной ЭВМ). После выдачи соответствующей команды оператор получает сообщение, указывающее имя текущего узла, и затем работа за терминалом в основном не отличается от работы за обычным терминалом ОСРВ. Если в удаленной ЭВМ работает система с многопользовательской защитой, необходимо после первоначальной выдачи команды RCL зарегистрироваться при помощи команды HELLO. Отметим, что при возврате к работе в локальной ЭВМ по команде LCL выдача команды BYE не обязательна, но рекомендуется по окончании работы её выполнять.

Наблюдается определенная аналогия между переключением терминала в подчиненный режим (по команде SET/SLAVE) и подключением терминала командами RCL и LCL. До выдачи команды RCL терминал, являясь не подчиненным терминалом в локальной ЭВМ, выступает в удаленной ЭВМ как виртуальное терминальное устройство (т. е. является устройством ввода-вывода, с которого не может поступить команда оператора). Выдача команды RCL переводит терминал в состояние передачи команд оператора в удаленную ЭВМ и исключает возможность выполнения команд в локальной ЭВМ. Это не отменяет возможности выполнения задачами в локальной ЭВМ операции ввода-вывода на терминале. Команда LCL возвращает терминал в исходное состояние (до выдачи RCL) Формат команды RCL:

RCL имя-узла,

где имя-узла имя удаленной ЭВМ.

Формат команды LCL:

LCL

8

ПРОГРАММНЫЕ СРЕДСТВА ПОВЫШЕНИЯ НАДЕЖНОСТИ ФУНКЦИОНИРОВАНИЯ СМ ЭВМ

ОСРВ включает следующие программные средства, позволяющие повысить надежность работы СМ ЭВМ:

подсистему регистрации ошибок оборудования;

программу-анализатор аварийного состояния системы (CDA);

пакет прикладных программ проверки работоспособности оборудования ППП «Эксперт»*;

программу проверки функционирования оборудования «Прораб»*;

подсистему резервирования дисков;

средства управления динамической памятью системы.

8.1. ПОДСИСТЕМА РЕГИСТРАЦИИ ОШИБОК ОБОРУДОВАНИЯ

Подсистема предназначена для сбора и обработки информации об ошибках оборудования, обслуживаемого системой.

Состав и взаимодействие элементов подсистемы. Подсистема состоит из трех задач:

регистрации ошибок оборудования (ERRLOG);

управления регистрацией (ELI);

генерации протокола ошибок (RPT).

Кроме того, к подсистеме относятся специальные подпрограммы управляющей программы, получающие информацию об ошибках непосредственно от драйверов уст-

* ППП «Эксперт» и программа «Прораб» подробно описываются в практическом руководстве [6]

ройств. Возможность регистрации ошибок оборудования выбирается пользователем при генерации системы.

При обнаружении ошибки оборудования драйвером устройства или управляющей программой управление передается специальным подпрограммам, которые создают в динамической памяти системы пакет данных, характеризующий данную ошибку. Созданные пакеты по мере накопления переписываются в файл на дисковом устройстве задачей ERRLOG.

Работа задачи ERRLOG регулируется с помощью управляющей информации, которую посылает ей задача ELI в виде пакетов данных специального вида, называемых *командными*. Затем на основе информации в файле, созданном задачей ERRLOG, задача RPT генерирует протокол регистрации ошибок оборудования — конечный результат сеанса работы подсистемы регистрации.

Типы ошибок оборудования, регистрируемые подсистемой. Подсистемой регистрируются четыре типа ошибок:

незапланированные прерывания;
ошибки внешних устройств;
тайм-ауты внешних устройств;
ошибки оперативной памяти.

Ошибки последнего типа регистрируются только в том случае, если в комплексе применяется оперативная память с архитектурной возможностью контроля четности.

Незапланированные прерывания возникают в системе, если в качестве вектора прерывания используется резервный вектор. Чаще всего такая ситуация создается из-за сбоя питания.

Ошибки внешних устройств фиксируются драйверами при выполнении операции ввода-вывода. При этом, как уже было упомянуто, создается пакет данных, включающий содержимое регистров контроллера устройства или другую зависящую от устройства информацию.

Различаются устранимые (корректируемые) и неустранимые (некорректируемые) ошибки устройств. *Устранимые* — это такие ошибки, которые может скорректировать драйвер устройства путем выполнения заданного количества повторных операций. *Неустранимые* ошибки вызывают ненормальное завершение операции ввода-вывода, так как драйвер, исчерпав лимит повторения операций на устройстве, не может её скорректировать. Следует подчеркнуть, что только появление неустранимых ошибок на устройстве говорит о его ненадежной работе. Однако большое число устранимых ошибок, особенно тех,

которые корректируются не с первой попытки, свидетельствует о возможном приближении сбоя.

Тайм-аут возникает на устройстве, когда запущенная на нем операция ввода-вывода не может завершиться за интервал времени, предусмотренный драйвером. По истечении указанного интервала времени фиксируется ошибка по тайм-ауту.

Ошибки оперативной памяти возникают в том случае, когда бит четности, запомненный с данными во время операции записи в память, не совпадает с битом четности этих данных, сформированном при чтении из памяти. Возможность регистрации ошибок памяти выбирается пользователем при генерации системы.

Ключи задачи ELI. Ключи этой задачи могут быть разбиты на четыре группы:

ключ регистрации;

ключи ограничения;

ключи назначения файла;

ключ вывода на терминал.

Ключ регистрации:

[спецф.]/LOG — по этому ключу начинается (или при использовании в отрицательном смысле останавливается) регистрация ошибок. Необязательный аргумент [спецф.] определяет спецификацию файла, в который выводятся данные об ошибках, собранные задачей ERRLOG. По умолчанию используется спецификация файла LB: [1,6] LOG.ERR.

Возможно использование подключей:

/—LIM — отмена ограничения числа регистрируемых ошибок. По умолчанию заранее определяется число ошибок на устройстве, по достижении которого ERRLOG перестает их регистрировать (например для диска —8);

/NV — создает новую версию файла регистрации ошибок. По умолчанию новые данные добавляются в текущий файл регистрации ошибок (если он существует).

Ключи ограничения. В задаче ELI насчитывается четыре ключа ограничения:

/LIM — по этому ключу определяется число регистрируемых ошибок на устройстве;

/HL:N — данный ключ устанавливает предел для числа неустранимых ошибок, регистрируемых на устройстве или устройствах; N изменяется от 0 до $(254)_{10}$, по умолчанию N=5;

/SL:N — это ключ устанавливает предел для числа устранимых ошибок, регистрируемых на устройстве или устройствах;

N изменяется от 0 до $(254)_{10}$, по умолчанию N = 8. Ключи /SL и /HL можно использовать в одной командной строке;

/RE — этот ключ сбрасывает в нуль счетчик операций ввода-вывода и счетчик числа ошибок на устройстве или устройствах. В командной строке может быть указано до 14 имен устройств.

Ключи назначения файла. Предусмотрено три таких ключа:

спецф./AP — по этому ключу содержимое специфицированного файла добавляется в конец текущего файла регистрации ошибок. При этом подсистема регистрации ошибок должна быть активна. Возможно использование подключа /DE, по которому специфицированный файл после присоединения удаляется;

[спецф.]/SW — по этому ключу содержимое текущего файла регистрации ошибок копируется в специфицированный файл, и регистрация переключается в этот файл. Возможно использование двух подключей:

/DE — текущий файл регистрации ошибок удаляется после завершения переписи в специфицированный файл;

/NV — создается новая версия специфицированного файла;

спецф./BA — ключ указывает имя резервного файла, который будет использоваться, если невозможен вывод данных в файл регистрации ошибок. По ключу автоматически создается новая версия специфицированного файла. Умолчанием для спецификации резервного файла является LB: [1,6] BACKUP.ERR.

Ключ вывода на терминал: /SH — ключ позволяет вывести на терминал, с которого активизирована задача ERRLOG, информацию о текущем состоянии регистрации ошибок в системе. В командной строке может быть указано до 14 имен устройств. Если же имена устройств не указаны, то выводится информация обо всех устройствах в системе, на которых возможна регистрация ошибок.

Формат командной строки задачи RPT:

[фсооб [/ключи]] = [фввод [/ключи]]

где фсооб — спецификация файла вывода протокола регистрации ошибок;
ключи — используется произвольный порядок ключей;
фввод — файл регистрации ошибок оборудования, созданный задачей ERRLOG.

В качестве вводного файла используется также универсальная библиотека модулей описания оборудования LB: [1,6] ERRLOG.ULB. Этот файл не указывается в командной строке.

Умолчания спецификаций файлов задачи RPT приведены ниже.

Поле	Файл протокола	Файл ввода
Имя устройства	SYO:	LB0:
Каталог	Текущий UIC	[1,6]
Имя файла	ERRREPORT	LOG
Тип файла	.LST	.ERR
Версия файла	Новая	Последняя

Ключи задачи RPT. Ключи этой задачи можно разбить на две группы:

ключи выбора пакетов из файла регистрации ошибок;
ключи форматирования выбранных пакетов.

Ключи выбора пакетов. В задаче RPT предусмотрено пять ключей выбора пакетов.

/DA: опр — ключ позволяет выбрать событие, основываясь на дате его появления. Здесь опр — определители, т. е. значения ключа специального вида. Возможны следующие определители:

P:N — выбор пакетов за предыдущие N дней;
R: начало:конец — задает период времени для выбора пакетов. Начальная и конечная даты задаются в стандартном формате ОСРВ. Символ * в качестве начальной даты означает все события вплоть до конечной, а в качестве конечной — все события от начальной даты;
T — выбираются пакеты, помеченные текущей датой;
Y — выбираются пакеты, помеченные вчерашней датой.
По умолчанию —/DA:R:*:*

/DE:(опр 1, ..., опр N) — ключ выбирает пакеты, связанные с именами устройств, специфицированными в определителях. Возможны следующие значения определителей:

A[LL] — все устройства;
DD — мнемоника имени устройства;
DDNN:— имя и номер устройства. Выбирает пакеты, относящиеся только к специфицированному устройству

По умолчанию —/DE:A

/P:(опр 1, ..., опр N) — ключ позволяет выбрать пакеты в соответствии с их номерами. Каждый определитель в данном случае имеет формат NNNN.NNN [:YYYY.YYY], определяющий диапазон номеров пакетов. Символ * позволяет выбрать все пакеты, начиная с указанного номера или до указанного номера. Номера требуемых пакетов можно получить, используя короткую форму протокола.

По умолчанию — P:*:*

/T:(опр 1, ..., опр N) — ключ осуществляет выбор в соответствии с типом содержимого пакетов. Возможны следующие определители:

A[LL] — все пакеты;

C[ONTROL] — командные пакеты;

E[RRORS] — пакеты, связанные с ошибками оборудования;

M[EMORY] — пакеты, связанные с ошибками памяти;

PE[RIPHERAL] — пакеты, связанные с ошибками внешних устройств;

PR[OCESSOR] — пакеты, связанные с ошибками процессора;

S[YSTEM-INFORMATION] — пакеты, связанные с ошибками, при выполнении некоторых операций системы (например, монтирование и демонтирование томов).

По умолчанию —/T:A

/V:метка — ключ позволяет выбирать пакеты, связанные с ошибками устройства, на котором создан том файловой структуры с данной меткой. Умолчания для ключа нет.

Ключи форматирования. В задаче RPT насчитывается три ключа форматирования:

/F:определитель — основной ключ форматирования выводного протокола. Возможные значения определителя:

B[RIEF] — генерирует короткую форму протокола. При этом в выводном протоколе для каждой ошибки отводится одна строка;

F[ULL] — генерирует полный протокол. При этом в выводном протоколе содержится вся информация из пакета регистрации ошибок;

N[ONE] — протокол не генерируется. Определитель может быть использован для проверки содержимого файла регистрации ошибок оборудования;

R[EGISTER] — протокол содержит только состояние регистров устройства в момент ошибки.

По умолчанию —/F:B

/W:опр — ключ определяет ширину строки протокола. Возможны следующие значения определителя:

N[ARROW] — узкий протокол, шириной 80 знаков;

W[IDE] — широкий протокол, шириной 132 знака.

По умолчанию — /W:W

/R:строка — данный ключ позволяет задаче RPT использовать заранее определенную строку из вышеописанных ключей. Заранее определенные строки ключей находятся в модуле управляющей программы PARSEM, который просматривается вначале, и в файле LB: [1,6] ERRDEFINE.CFB. Пользователь может создавать свои строки ключей и включать их в указанный файл.

Системой определяются (в модуле PARSEM) следующие строки ключей:

```
DAY=/F:F /DA:T
WEEK=/DA:P:7
MONTH=/DA:P:31
SYSTEM    все значения ключей по умолчанию.
```

8.2. ПРОГРАММА-АНАЛИЗАТОР АВАРИЙНОГО СОСТОЯНИЯ СИСТЕМЫ (CDA)

Программа CDA используется для анализа причин аварийного завершения работы системы. Входной для нее является информация, сохраненная системой на внешнем устройстве в момент аварии. Запуск CDA аналогичен запуску любой другой системной обслуживающей программы ОСРВ.

Формат командной строки CDA:

[файл-расп.[/кл]] [,дв.выв.файл/кл] = [файл-опр.-символов /STB,] дв.вв.файл [/кл]

где файл-расп.— файл распечатки, выводимой CDA. Наличие этого файла не обязательно;

дв.выв.файл — этот файл является копией двоичной информации, сохраненной на внешнем устройстве в момент аварии системы. Указанный файл используется для накопления информации об аварийном состоянии системы. Однажды созданный, он может использоваться впоследствии как вводной файл CDA. Файл не является обязательным;

файл-опр.-символов/STB — файл, содержащий значения глобальных символов системы, в которой произошла авария. Ключ /STB является неотъемлемой частью спецификации файла. Наличие данного файла не обязательно;

дв.вв.файл — двоичный вводной файл. Здесь может использоваться либо имя устройства сохранения, либо спецификация двоичного выводного файла, созданного задачей CDA ранее. Во втором случае в командной строке должен отсутствовать двоичный выводной файл;

/кл — каждый ключ связан с определенным файлом и не может быть использован без спецификации файла или с другими файлами. Наличие ключей не обязательно.

Умолчания, используемые в командной строке CDA, сведены в табл. 8.1.

Таблица 8.1

Умолчания спецификаций файлов задачи CDA

Элемент командной строки	Устройство	Каталог	Имя файла	Тип файла
Файл распечатки	SY:	Текущий UIC	Нет	LST
Двоичный выводной файл	SY:	Текущий UIC	Нет	CDA
Файл определения символов	SY:	Текущий UIC	OCPBCM	STB
Двоичный вводной файл	SY:	Текущий UIC	Нет	CDA

Ключи программы CDA. Ключи, используемые в командной строке CDA, можно разбить на две группы:

ключи анализа, управляющие распечаткой информации, выбранной из двоичного файла. Все ключи анализа используются с двоичным вводным файлом;

ключи функционирования, задающие параметры работы CDA.

Ключи анализа. Ниже перечисляются ключи анализа и дается краткая характеристика их использования.

/ACT(/ATL) — по ключу распечатывается содержимое блоков управления задачей TCB (см. приложение) для всех активных задач.

/ADV — по ключу распечатывается содержимое всех блоков управления для всех устройств в системе.

/ALL — по ключу печатается: системная информация, информация об активных задачах, заголовки установленных в системе задач, информация о разделах, информация об устройствах, очередь запросов по времени, содержимое динамической памяти системы.

/CLI (/CPB) — распечатывает содержимое всех блоков описания CLI (CPB).

/CLQ — распечатывает очередь запросов по времени.

/DEV(/DCB, /SCB, /UCB) — распечатывает содержимое блоков управления всех активных устройств в системе. Ключ /DEV печатает все блоки для всех активных устройств; ключ /DCB — только блоки DCB; ключ /SCB — блоки SCB и ключ /UCB — блоки UCB.

/DUMP:A:B (/DMP:A:B) — распечатывает содержимое физической памяти, начиная с 18-разрядного адреса A до адреса B.

/HDR — по ключу распечатываются заголовки всех задач, находящихся в памяти.

/PCB (/PAR) — распечатывает содержимое всех блоков управления разделами (PCB).

/POOL — содержимое динамической памяти системы печатается в восьмеричном символьном коде и в коде RADIX-50.

/—SYS — ключ позволяет подавить печать системной информации, обычно присутствующей в распечатке. Системная информация включает содержимое регистров процессора, общую характеристику системы в момент аварии, содержимое системной общей области SYSCM, характеристики динамической памяти системы, таблицу логических назначений устройств и др.

/TASK=имя [:A:B] (/TAS,/TSK) — рассматривается содержимое виртуального адресного пространства указанной задачи, начиная с 16-разрядного адреса A до

адреса B. Если диапазон адресов не указан, печатается все виртуальное адресное пространство задачи.

/STD — распечатывается содержимое TCB для всех установленных задач.

Ключи функционирования. Ниже приводится краткая характеристика этих ключей.

/BL:N — ключ указывает, что считывание информации из двоичного вводного файла должно начинаться с блока номер N. Ключ игнорируется для недисковых устройств, по умолчанию N=1. Используется с двоичным вводным файлом.

/DENS:N (/DENS:HIGH, /DENS:LOW) — задает плотность записи на магнитной ленте, содержащей двоичный вводной файл. Аргумент LOW означает 32 бит/мм, HIGH —64 бит/мм. По умолчанию — LOW. Используется с двоичным вводным файлом.

/EXIT:N — по ключу работа программы CDA завершается после обнаружения N ошибок, по умолчанию N=1. В случае отсутствия ключа CDA всегда работает до конца. Используется с файлом распечатки.

/LIMIT:N — ключ ограничивает число страниц в файле распечатки. После создания N страниц программа прекращает работу и печатает соответствующее сообщение на терминале: по умолчанию N=300. Используется с файлом распечатки.

/LINES:N — позволяет задать число строк в странице распечатки, по умолчанию N=60. Используется с файлом распечатки.

/MEMSIZ:N — определяет размер двоичного выводного файла. При этом создается файл длиной 4×N блоков, в который переписывается из двоичного вводного файла 2×N Кбайт; по умолчанию N=124. Используется с двоичным выводным файлом.

/KMR — может быть использован в том случае, если есть опасность, что в процессе работы CDA виртуальные системные адреса будут неверно преобразованы в физические вследствие ошибочной информации, содержащейся в некоторых регистрах адреса страниц (PAR) [1]. В этом случае CDA будет использовать стандартные способы отображения. По умолчанию CDA использует текущие регистры адреса страниц. Используется с двоичным вводным файлом.

/— SP — запрещение системного вывода. В противном случае (по умолчанию) по завершению создания на устройстве файла распечатки будет осуществлен вывод на

печать с помощью программы системного вывода. Используется с файлом распечатки.

/STB — описание ключа дано при определении формата командной строки программы CDA.

8.3. ПОДСИСТЕМА РЕЗЕРВИРОВАНИЯ ДИСКОВ

Подсистема обеспечивает одновременную запись информации на несколько однотипных дисковых накопителей из некоторого набора, называемого *набором резервированных дисков*. Задача пользователя связана через логический номер (LUN) с одним так называемым базовым устройством из набора.

Чтение информации для задачи, использующей резервирование, начинается с эталонного устройства (по умолчанию оно совпадает с базовым) и в случае аппаратной ошибки продолжается на других резервированных дисках. Желательно, чтобы на каждом из резервированных дисков одинаковые блоки были признаны дефектными. Этого можно добиться, используя специальную функцию подсистемы.

В состав подсистемы резервирования дисков входят: задача, управляющая резервированием (RSV), виртуальный драйвер резервированного диска и задача, корректирующая файл дефектных блоков (UPD).

Возможность резервирования дисков выбирается при генерации системы. Для этого на первой фазе генерации надо утвердительно ответить на следующий вопрос:

```
RESERVATION OF DISK'S (Y/N)
РЕЗЕРВИРОВАНИЕ ДИСКОВ? (ДА/НЕТ)
```

Кроме этого, в первой фазе генерации на этапе выбора периферийных устройств относительно каждого дискового устройства задается вопрос о максимальном количестве резервированных дисков данного типа. Если это число отлично от нуля сразу для нескольких типов дисков, то будут построены резидентные виртуальные драйверы резервированных дисков соответствующих типов. В противном случае драйвер может быть загружаемым. Тогда будет задан вопрос:

```
LOADABLE RESERVATION DRIVER OF DISK'S? (Y/N)
ЗАГРУЖАЕМЫЙ ДРАЙВЕР РЕЗЕРВИРОВАННОГО ДИСКА? (ДА/НЕТ)
```

Утвердительный ответ (Y) приводит к построению загружаемого драйвера.

Во второй фазе генерации должны быть построены задачи RSV и UPD. Для этого в перечне необходимых привилегированных задач надо указать RSV. До создания задачи RSV надо отредактировать ее командный файл построения, т. е. внести в соответствующую таблицу имена виртуальных устройств для резервируемых типов дисков. Эти имена имеют вид VX, где X — второй символ в имени устройства. Например, для диска DK соответствующим именем виртуального устройства будет VK. Чтобы сделать возможным резервирование дисков DK, DP и DM, нужно выполнить следующую корректировку файла построения задачи RSV:

```
GBLPAT=RSV:DEVTAB:45 526 (VK)
GBLPAT=RSV:DEVTAB+2:50 126 (VP)
GBLPAT=RSV·DEVTAB+4:46 526 (VM)
```

Если выбран загружаемый виртуальный драйвер, то будет построен VXDRV.TSK (здесь X — также второй символ в имени устройства), который можно загрузить командой LOA VK: (для дисков DK).

Формат команды резервирования RSV:

```
RSV DEV·[/SWI I/SWI .. I/SW] [,DEV:] [/SW] .
или
RSV /SW
```

где DEV — имя резервированного диска;
SW — ключ команды.

Ниже приводится описание возможностей команды RSV.

Создание набора резервированных устройств. Набор устройств создается по команде:

```
RSV DEV1:/BAS, DEV2: ...
```

где все диски должны быть одного типа. Ключ /BAS является обязательным и объявляет то устройство, с которым он связан, базовым. Оно же объявляется эталонным.

Возможно использование следующих подключей:

/COPY — при включении устройства в набор резервированных устройств на него копируются данные с эталонного устройства. Подключ действует по умолчанию;

/VFY — выполняется проверка сделанной копии устройства;

/NOCOPY — запрещается копирование на вновь включаемое устройство.

Расширение набора резервированных устройств. То же, что в предыдущем без ключа /BAS. Возможно использование тех же подключей.

Проверка идентичности двух резервированных дисков. Выполняется по команде:

RSV DEV1./CMP, DEV2:

В случае несовпадения информации на устройствах выдается сообщение об ошибке.

Назначение эталонного устройства. Выполняется по команде:

RSV DEV:/ORIG

При этом заменяется эталонное устройство. Устройство должно быть предварительно включено в набор резервированных дисков.

Поиск дефектных блоков в наборе резервированных устройств. Выполняется по команде:

RSV DEV./BAD

где DEV — имя базового устройства.

Устанавливается режим поиска дефектных блоков на всех устройствах набора. После установки этого режима надо запустить программу поиска дефектных блоков BAD на базовое устройство. В результате на всех устройствах набора создается объединенный описатель дефектных блоков. Режим отменяется по команде:

RSV DEV:/NOBAD

Исключение из набора резервированных устройств. Выполняется по команде:

RSV DEV:/REM

где DEV — имя устройства, исключаемого из набора резервированных дисков.

Базовое устройство по этой команде не исключается из набора, на нем только прекращаются операции ввода-вывода.

Уничтожение набора резервированных устройств. Выполняется по команде:

RSV DEV:/UNRSV

где DEV — имя базового устройства набора.

Получение информации о резервированных устройствах. Выполняется по команде:

RSV /DISPLAY

При этом на терминале печатается информация о возможных типах резервированных устройств, а также о состоянии имеющихся наборов резервированных дисков.

Печать списка устройств, для которых возможно резервирование. Выполняется по команде:

```
RSV /DEV
```

При этом на терминале печатается список виртуальных имен устройств, соответствующих тем типам дисков, для которых при генерации включена возможность резервирования.

8.4. СРЕДСТВА УПРАВЛЕНИЯ ДИНАМИЧЕСКОЙ ПАМЯТЬЮ СИСТЕМЫ

Возможность управления динамической памятью системы (пулом) выбирается при генерации системы. Для этого необходимо утвердительно (Y) ответить на следующий вопрос первой фазы генерации:

```
POOL MANAGEMENT ? [Y/N]
УПРАВЛЕНИЕ ПУЛОМ ? (Да/нет)
```

Основным элементом системы является привилегированная задача PMT. Она инициируется через определенный интервал времени и выполняет различные действия в зависимости от состояния динамической памяти. Фиксируются два основных состояния:

низкое, когда количество свободной динамической памяти меньше установленного предела (по умолчанию 100 байт) или возникает отказ в выделении динамической памяти;

высокое, когда объем свободной динамической памяти превышает установленный предел.

При низком состоянии пула задача PMT:

запрещает свою выгружаемость;

устанавливает интервал своего перезапуска, равный 20 с;

запрещает регистрацию непривилегированных пользователей;

отвергает команды INS, RUN и REM, изданные с непривилегированных терминалов;

посылает специальные сообщения на все терминалы, зарегистрированные в системе, и др.

При высоком состоянии пула задача РМТ:

разрешает свою выгружаемость;

устанавливает интервал своего перезапуска, равный 1 мин;

включает контроль за использованием пула.

При экстремальной фрагментации, когда максимальная непрерывная область динамической памяти меньше 84 байт, программа РМТ осуществляет следующие действия:

запрещает выполнение других задач, в том числе задачи TKTN;

печатает на консольном терминале список задач, рекомендуемых для аварийного завершения, и др.

* * *

Операционная система ОСРВ находится в процессе развития.

Следующая версия системы — 3.1, полностью реализующая возможности версии 3.0, которая описана в настоящей книге, включит новые системы программирования, дополнительные средства управления системой, пакеты прикладных программ.

ЛИТЕРАТУРА

1 СМ ЭВМ: Комплексирование и применение/Под ред. Н. Л. Прохорова.—2-е изд., перераб. и доп.—М.: Финансы и статистика, 1986.—304 с.

2. *Васильев Г П., Егоров Г А., Щербина Н. Н.* Программное обеспечение сетей СМ ЭВМ.—М.: Финансы и статистика, 1983.—86 с

3. *Васильев Г П., Горский В. Е., Шяудкулис В. И., Саух Н. Н* Программное обеспечение неоднородных распределенных систем.—М.. Финансы и статистика, 1986.—163 с.

4. Государственная система промышленных приборов и средств автоматизации: Система малых ЭВМ. Программное обеспечение.—Т 5.—Вып. 4.—М., ЦНИИТЭИприборостроения, 1984.—С. 28—30, 32—33.

5. *Вигдорчик Г В., Воробьев А. Ю., Праченко В. Д.* Программирование на Ассемблере для СМ ЭВМ.—М.: Финансы и статистика, 1983.—252 с.

6. *Столяр Л Н., Шапошников В. А.* Средства проверки работоспособности оборудования СМ ЭВМ.—М.: Финансы и статистика, 1986.—142 с.

7 *Бирюков В. В., Рыбаков А. В., Шакула Ю. П* Введение в систему программирования ОСРВ.—М.: Финансы и статистика, 1986.—192 с.

КОДЫ ОШИБОК

В данном приложении приводятся числовые значения кодов состояний директив, кодов состояний завершения ввода-вывода для всех функций ввода-вывода и кодов общих функций директивы QIO⊙ Для каждого кода указывается символическое имя в формате IO.XXX, IE.XXX или IS.XXX, а также десятичное значение и краткое описание ошибки или функции.

1.1. КОДЫ СОСТОЯНИЯ ВВОДА-ВЫВОДА

В табл. П.1 — П.6 приводятся коды ошибочного и нормального завершения ввода-вывода, которые передаются в блоке состояния ввода-вывода. Если в директиве .MCALL указать в качестве аргумента IOERR⊙ и включить макрокоманду IOERR⊙ в программу, то на перечисленные ниже коды можно ссылаться по символическим именам.

Таблица П.1

Коды ошибок в блоке состояния ввода-вывода для всех функций

Имя	Десятичное значение	Описание ошибки
IE.BAD	—01	Неверный параметр
IE.IFC	—02	Недопустимый код функции
IE.DNR	—03	Устройство не готово
IE.VER	—04	Ошибка четности на устройстве
IE.ONP	—05	Недопустимая подфункция для оборудования
IE.SPC	—06	Недопустимый адрес буфера
IE.DNA	—07	Устройство не присоединено
IE.DAA	—08	Устройство уже присоединено
IE.DUN	—09	Устройство неприсоединяемое
IE.EOF	—10	Обнаружен конец файла
IE.EOV	—11	Обнаружен конец тома
IE.WLK	—12	Попытка записи на устройство, защищенное от записи
IE.DAO	—13	Потеря данных
IE.SRE	—14	Ошибка при посылке и получении данных
IE.ABO	—15	Операция прекращена
IE.PRI	—16	Нарушение привилегий
IE.RSU	—17	Разделяемый ресурс занят
IE.OVR	—18	Недопустимый запрос на считывание перекрытия

Продолжение

Имя	Десятичное значение	Описание ошибки
IE.BYT	—19	Нечетный счетчик байтов или буфер не на границе слова
IE.BLK	—20	Недопустимый номер логического блока
IE.BBE	—56	Дефектный блок на устройстве
IE.STK	—58	Переполнение стека (FCS или FCP)
IE.FHE	—59	Неисправимая ошибка оборудования на устройстве
IE.EOT	—62	Обнаружен конец ленты
IE.OFL	—65	Устройство выключено
IE.BCC	—66	Ошибка циклического контроля
IE.NDR	—72	Не хватает динамической памяти
IE.TMO	—95	Произошел тайм-аут

Таблица П.2

Коды ошибок в блоке состояния ввода-вывода, передаваемые процессором управления файлами FCP

Имя	Десятичное значение	Описание ошибки
IE.NOD	—23	Нет свободной динамической памяти
IE.DFU	—24	Устройство переполнено
IE.IFU	—25	Индексный файл переполнен
IE.NSF	—26	Нет такого файла
IE.LCK	—27	Блокировано от доступов чтение/запись
IE.HFU	—28	Заголовок файла переполнен
IE.WAC	—29	Файл открыт для записи
IE.CKS	—30	Ошибка контрольной суммы в заголовке файла
IE.WAT	—31	Неверный формат блока управления атрибутами
IE.RER	—32	Ошибка чтения при работе задачи FCP
IE.WER	—33	Ошибка записи при работе задачи FCP
IE.ALN	—34	Файл уже открыт по данному логическому номеру устройств (LUN)
IE.SNC	—35	Ошибка номера файла в идентификаторе файла
IE.SQC	—36	Ошибка последовательного номера файла в идентификаторе файла
IE.NLN	—37	Файл недоступен по данному логическому номеру устройства (LUN)
IE.CLO	—38	Файл неверно закрыт
IE.DUP	—57	Попытка ввести в каталог запись о файле с уже существующим именем, типом и версией
IE.BVR	—63	Неверный номер версии
IE.BHD	—64	Неверный заголовок файла
IE.EXP	—75	Дата удаления файла не достигнута
IE.BTF	—76	Неверный формат ленты
IE.ALC	—84	Ошибка при размещении файла
IE.ULK	—85	Ошибка при разблокировании
IE.WCK	—86	Ошибка контроля записи

Таблица П.3

Коды ошибок в блоке состояния ввода-вывода, передаваемые системой управления файлами FCS

Имя	Десятичное значение	Описание ошибки
IE.NBF	—39	Нет памяти для размещения буфера файла (OPEN)
IE.RBG	—40	Неверный размер записи
IE.NBK	—41	Файл превышает выделенное пространство на диске
IE.ILL	—42	Неверная операция в блоке FDB
IE.BTP	—43	Неверный тип записи
IE.RAC	—44	Неверные биты доступа к записям
IE.RAT	—45	Неверные биты атрибутов записи
IE.RCN	—46	Слишком большой номер записи
IE.2DV	—48	Операция переименования выполняется на двух разных устройствах (RENAME)
IE.FEX	—49	Новое имя файла уже использовано (RENAME)
IE.BDR	—50	Неверный файл каталога
IE.RNM	—51	Переименование невозможно
IE.BDI	—52	Неверный синтаксис в имени каталога
IE.EOP	—53	Файл уже открыт
IE.BNM	—54	Неверное имя файла
IE.BDV	—55	Неверное имя устройства
IE.NFI	—60	Не указан идентификатор файла
IE.ISQ	—61	Неверная последовательность операций
IE.NNC	—77	Нестандартный формат счетчика байтов на магнитной ленте

Таблица П.4

Коды ошибок в блоке состояния ввода-вывода для терминалов

Имя	Десятичное значение	Описание ошибки
IE.IES	—82	Неверная ESC-последовательность
IE.PES	—83	Частичная ESC-последовательность

Таблица П.5

Коды нормального завершения в блоке состояния ввода-вывода

Имя	Десятичное значение	Описание ошибки
IS.PND	+00	Запрос ввода-вывода в состоянии ожидания
IS.SUC	+01	Нормальное завершение
IS.RDR	+02	Успешное завершение операции чтения физического блока для гибкого диска
IS.DAO	+02	Успешное завершение, но произошло переполнение данных

Таблица П.6

Коды нормального завершения для терминала

Имя	Десятичное значение	Описание кода
IS.CR	Байт 0:1 Байт 1:15	Завершение с возвратом каретки
IS.ESC	Байт 0:1 Байт 1:33	Завершение с символом ESC
IS.CC	Байт 0:1 Байт 1.3	Завершение с символом CTRL/C
IS.ESQ	Байт 0:1 Байт 1:233	Завершение ESC последовательностью ESC
IS.PES	Байт 0:1	Завершение с частичной последовательностью ESC
IS.EOT	Байт 0:1 Байт 1:4	Завершение с кодом EOT (режим ввода блока)
IS.TAB	Байт 0:1 Байт 1:11	Завершение с кодом TAB (горизонтальная табуляция)
IS.TMO	2	Был тайм-аут на запрос

1.2. КОДЫ СОСТОЯНИЯ ДИРЕКТИВ

В табл. П.7 и П.8 приводятся коды ошибочного и нормального завершения директивы QIO, которые передаются в слове состояния директивы DSW. К этому слову ссылаются по символическому имени ⊙DSW. Если в директиве .MCALL указать в качестве аргумента DRERR⊙ и включить макрокоманду DRERR⊙ в программу, то на перечисленные ниже коды можно ссылаться по символическим именам.

Таблица П.7

Коды ошибок в слове состояния директивы

Имя	Десятичное значение	Описание ошибки
IE.UPN	—01	Недостаточно динамической памяти
IE.INS	—02	Задача не установлена
IE.PTS	—03	Раздел слишком маленький для задачи
IE.UNS	—04	Недостаточно динамической памяти при посылке данных
IE.LUN	—05	Не назначен логический номер устройства (LUN)
IE.HWR	—06	Драйвер устройства отсутствует в памяти
IE.ACT	—07	Задача не активна
IE.ITS	—08	Директива не соответствует состоянию задачи
IE.FIX	—09	Задача уже фиксирована или не фиксирована в памяти (в зависимости от запроса)
IE.CKP	—10	Запрашиваемая задача невыгружаемая
IE.TCH	—11	Задача выгружаемая
IE.RBS	—15	Буфер получения данных слишком маленький

Продолжение

Имя	Десятичное значение	Описание ошибки
IE.PRI	—16	Нарушение привилегий
IE.RSU	—17	Ресурс занят
IE.NSW	—18	Нет области для обмена
IE.ILV	—19	Указан неверный вектор устройства
IE.AST	—80	Неверная директива в подпрограмме AST
IE.MAP	—81	Указанное отображение недопустимо
IE.IOP	—83	Окно имеет выполняющийся ввод-вывод
IE.ALG	—84	Ошибка выравнивания
IE.WOV	—85	Переполнение при размещении адресного окна
IE.NVR	—86	Неверный ID района
IE.NVW	—87	Неверный ID адресного окна
IE.ITP	—88	Неверный параметр TI
IE.IBS	—89	Неверный размер буфера при посылке (больше 255)
IE.LNL	—90	LUN блокирован для использований
IE.IUI	—91	Неверный код UIC
IE.IDU	—92	Неверное устройство
IE.ITI	—93	Неверные параметры времени
IE.PNS	—94	В системе нет указанного раздела или района
IE.IPR	—95	Приоритет задачи больше 250
IE.ILU	—96	Недопустимый номер LUN
IE.IFF	—97	Недопустимый номер флага события (больше 64)
IE.ADP	—98	Недопустимое размещение блока DPB (вне границ памяти задачи)
IE.SDP	—99	Недопустимый идентификационный код директивы (DIC) или размер блока DPB

Таблица П.8

Коды нормального завершения в слове состояния директивы

Имя	Десятичное значение	Описание кода
IS.CLR	0	Флаг события сброшен (для директивы CLEF⊙)
IS.SET	2	Флаг события установлен (для директивы CLEF⊙)
IS.SPD	2	Задача приостановлена

1.3. КОДЫ ОБЩИХ ФУНКЦИЙ ДИРЕКТИВЫ QIO

Символические имена кодов общих функций директивы QIO⊙ определены в макрокоманде FILIO⊙ (табл. П. 9).

Таблица П.9

Коды общих функций директивы QIO

Имя	Модификатор*	Код*	Описание кода
IO.KIL	012	000	Остановить ввод-вывод
IO.RDN	022	000	Запустить ввод-вывод
IO.UNL	042	000	Удалить драйвер устройства
IO.LTK	050	000	Загрузить файл образа задачи
IO.RTK	060	000	Записать файл образа задачи
IO.SET	030	000	Установить характеристики
IO.WLB	000	001	Записывать логический блок
IO.RLB	000	002	Читать логический блок
IO.LOV	010	002	Загрузить перекрытие
IO.ATT	000	003	Присоединить устройство к задаче
IO.DET	000	004	Отсоединить устройство от задачи
IO.FNA	000	011	Найти имя файла в каталоге
IO.RNA	000	013	Удалить имя файла из каталога
IO.ENA	000	014	Ввести имя файла в каталог
IO.CLN	000	007	Закрыть файл по данному LUN
IO.ULK	000	012	Разблокировать блок
IO.ACR	000	015	Разрешить доступ для чтения
IO.ACW	000	016	Разрешить доступ для записи
IO.ACE	000	017	Разрешить доступ для расширения
IO.DAC	000	020	Прекратить доступ к файлу
IO.RVB	000	021	Читать виртуальный блок
IO.WVB	000	022	Записывать виртуальный блок
IO.EXT	000	023	Расширить файл
IO.CRE	000	024	Создать файл
IO.DEL	000	025	Удалить файл
IO.RAT	000	026	Читать атрибуты файла
IO.WAT	000	027	Записать атрибуты файла

* Модификатор и код восьмеричные.

ПРИЛОЖЕНИЕ 2

ФОРМАТ ИНДЕКСНОГО ФАЙЛА

Индексный файл ([0,0]INDEXF SYS) тома с файловой структурой OCPB состоит из блоков заголовков всех файлов на дисковом устройстве.

Ниже приводятся виртуальные номера блоков и названия элементов индексного файла.

Виртуальный номер блока	Элемент индексного файла
1	Блок начального загрузчика
2	Собственный блок тома
3	Карта планирования индексного файла (N блоков. Значение N хранится в собственном блоке тома)

3+N	Заголовок индексного файла
3+N+1	Заголовок файла распределения пространства диска
3+N+2	Заголовок файла дефектных блоков
3+N+3	Заголовок файла главного каталога файлов MFD
3+N+4	Заголовок файла для временной выгрузки на диск
3+N+5	Заголовок пользовательского файла 1
3+N+6	Заголовок пользовательского файла 2
.	
.	
.	
3+N+4+K	Заголовок пользовательского файла K

2.1. БЛОК НАЧАЛЬНОГО ЗАГРУЗЧИКА

На диске с файловой структурой ОСРВ имеется блок из 256 слов с логическим номером 0. Этот блок содержит или подпрограмму начальной загрузки или сообщение оператору о том, что том не содержит средств для начальной загрузки. Подпрограмма начальной загрузки загружает в память образ системы начиная с заранее определенного блока на диске.

2.2. СОБСТВЕННЫЙ БЛОК

Собственный блок тома содержит информацию, идентифицирующую том. Этот блок расположен или в блоке с логическим номером 1, или в любом блоке с номером, кратным 256.

Имена полей в собственном блоке могут быть определены как локальные или как глобальные в зависимости от аргумента макрокоманды HMBOF.

Формат собственного блока тома приведен в табл. П.10.

Таблица П.10

Формат собственного блока тома

Размер поля (в байтах)	Содержимое поля	Смещение
2	Размер карты планирования индексного файла	H.IBSZ
4	Адрес карты планирования индексного файла (логический номер блока)	H.IBLB
2	Максимально допустимое число файлов на томе	H.FMAX
2	Единица распределения пространства диска	H.SBCL
2	Тип диска	H.DVTY
2	Уровень структуры	H.VLEV
12	Имя тома (12 знаков в символьном коде)	H.VNAM
4	Резервируется	—
2	Код идентификации UIC владельца тома	H.VOWN
2	Код защиты тома	H.VPRO
2	Характеристики тома	H.VCHA

Продолжение

Размер поля (в байтах)	Содержимое поля	Смещение
2	Слово защиты файла по умолчанию	H.DFPR
6	Резервируется	—
1	Число указателей извлечения в блоке указателей извлечения по умолчанию	H.WISZ
1	Число блоков для расширения файлов по умолчанию	H.FIEX
1	Число входов в каталоге LRU	H.LRUC
11	Резервируется	—
2	Контрольная сумма слов 0—28	H.CHK1
14	Дата и время создания	H.VDAT
100	Метка заголовка тома (не используется)	—
82	Специальная системная информация (не используется)	—
254	Относительная таблица тома (не используется)	—
2	Контрольная сумма собственного блока (слова от 0 до 255)	H.CHK2

2.3. КАРТА ПЛАНИРОВАНИЯ ИНДЕКСНОГО ФАЙЛА

Карта планирования индексного файла отражает использование блоков для заголовков файлов в индексном файле. Карта планирования содержит бит на каждый блок заголовка файла в индексном файле. Биты в карте планирования и блоки в индексном файле нумеруются в полном соответствии друг другу.

Значения бита:

0 — указывает, что соответствующий биту блок заголовка свободен, процессор управления файлами FCP может использовать этот блок под заголовок создаваемого файла;

1—указывает, что блок заголовка занят, он является заголовком файла.

2.4. ЗАГОЛОВКИ СТАНДАРТНЫХ ФАЙЛОВ НА ТОМЕ С ФАЙЛОВОЙ СТРУКТУРОЙ ОСРВ

При создании файловой структуры ОСРВ на томе создаются заголовки следующих пяти стандартных файлов:

заголовок индексного файла — стандартный заголовок, связанный с индексным файлом;

заголовок файла распределения пространства диска — заголовок файла, который используется для управления назначением пространства на диске файлам;

заголовок файла дефектных блоков — дефектные блоки, обнаруженные аппаратно или программно объединяются в файл с этим заголовком (если такие блоки окажутся на диске);

заголовок файла главного каталога — файл главного каталога файлов содержит записи для индексного файла, для файла распределения пространства диска, для файла дефектных блоков, для самого

файла главного каталога MFD, для файла временной выгрузки и для всех каталогов пользовательских файлов UFD;

заголовок файла для временной выгрузки — блок идентифицирует файл, предназначенный в качестве области выгрузки для всех задач, временно выгружаемых на диск из раздела.

В ОСРВ задачи могут иметь область выгрузки и в самом образе задачи. Оставшиеся блоки индексного файла отводятся под заголовки пользовательских файлов.

ПРИЛОЖЕНИЕ 3

ФОРМАТ БЛОКА ЗАГОЛОВКА ФАЙЛА

Формат блока заголовка файла приведен в табл. П.11. После таблицы приводится подробное описание областей блока заголовка. Имена смещений могут быть определены или как локальные, или как глобальные: в зависимости от аргументов макрокоманды FHDOF⊙

Таблица П.11

Блок заголовка файла

Размер поля (в байтах)	Содержимое поля	Смещение
	Область заголовка	
1	Смещение до области идентификации (в словах)	H.IDOF
1	Смещение до области планирования (в словах)	H.MPOF
2	Номер файла.	H.FNUM
2	Последовательный номер файла	H.FSEQ
2	Уровень структуры и номер системы	H.FLEV
—	Смещение до кода идентификации владельца файла, состоящего из номера члена и номера группы	H.FOWN
1	Номер члена	H.PROG
1	Номер группы	H.PROJ
2	Код защиты файла	H.FPRO
1	Контролируемые пользователем характеристики файла	H.UCHA
1	Контролируемые системой характеристики файла	H.SCHA
32	Пользовательские атрибуты файла	H.UFAT
—	Размер (в байтах) области заголовка в блоке заголовка файла	S.HDHD
	Область идентификации	
6	Имя файла	I.FNAM
2	Тип файла	I.FTYP
2	Номер версии файла	I.FVER
2	Номер ревизии	I.RVNO
7	Дата ревизии	I.RVDT
6	Время ревизии	I.RVTI
7	Дата ревизии	I.CRDT
6	Время создания	I.CRTI
7	Дата уничтожения	I.EXDT
1	Для выравнивания на границу слова	
—	Размер (в байтах) области идентификации в блоке заголовка файла	S.IDHD

Продолжение

Размер поля (в байтах)	Содержимое поля	Смещение
	Область планирования	
1	Номер сегмента расширения	M.ESQN
1	Относительный номер тома расширения (не используется)	M.ERVN
2	Номер файла расширения	M.EFNU
2	Последовательный номер файла расширения	M.EFSQ
1	Размер (в байтах) поля счетчика блоков в указателе извлечения	M.CTSZ
1	Размер (в байтах) поля логического номера блока в указателе извлечения	M.LBSZ
1	Слова указателей извлечения, используемые в области планирования	M.USE
1	Максимальное число слов для указателей извлечения, имеющихся в области планирования	M.MAX
—	Начальный указатель извлечения	M.RTRV
—	Размер (в байтах) области планирования в блоке заголовка файла	S.MPHD
	Слово контрольной суммы	
2	Контрольная сумма слов от 0 до 255	H.CKSM

Примечание Слово контрольной суммы является последним словом блока заголовка файла. Указатели извлечения занимают пространство от конца области планирования до слова контрольной суммы.

3.1. ОБЛАСТЬ ЗАГОЛОВКА

В области заголовка блока заголовка файла содержится следующая информация:

смещение до области идентификации — слово 0, биты 0—7; этот байт содержит смещение до начала области идентификации относительно начала блока заголовка файла; смещение равно числу слов от начала блока заголовка до области идентификации;

смещение до области планирования—слово 0, биты 8—15; этот байт содержит смещение до начала области планирования относительно начала блока заголовка файла; смещение равно числу слов от начала блока заголовка до области планирования;

номер файла — это номер блока заголовка этого файла в индексном файле (при условии, что блоки в индексном файле нумеруются с 1, и первые 5 блоков отданы под стандартные заголовки),

последовательный номер файла — номер использования данного блока заголовка. Номер файла и последовательный номер файла составляют идентификационный номер (идентификатор) файла. Этот номер меняется при использовании того же блока заголовка для другого файла;

уровень структуры — идентифицирует систему, которая создала файл; значение [1,1] связано со всеми текущими томами с файловой структурой ОСРВ;

информация о владельце файла — это слово содержит номер группы и номер члена, которые составляют код идентификации UIC для файла; допустимый диапазон от [1, 1] до [377, 377]

код защиты файла — это слово указывает, кто может иметь доступ к файлу и в каком режиме; при создании файла пользователь указывает желаемый код защиты для файла;

характеристики файла — слово, определяющее состояние файла. Байт 0 определяет характеристики, контролируемые пользователем:

UC.CON = 200 — логически непрерывный файл;

UC.DLK = 100 — неверно закрытый файл.

Байт 1 определяет характеристики, контролируемые системой:

SC.MDL = 200 — файл отмечен для удаления;

SC.BAD = 100 — дефектный блок данных в файле;

пользовательские атрибуты файла — эта область состоит из 16 слов: первые 7 слов являются копией первых 7 слов блока описания файла FDB, если файл открыт; оставшиеся 9 слов области системой управления вводом-выводом FCS не используются.

3.2. ОБЛАСТЬ ИДЕНТИФИКАЦИИ

В области идентификации в блоке заголовка файла содержится следующая информация:

имя файла — владелец файла указывает имя файла до 9 знаков в коде RADIX-50, неиспользованная часть поля заполняется нулями;

тип файла — это слово содержит тип файла в коде RADIX-50;

номер версии файла — это слово содержит двоичный номер версии файла, указанный при создании файла;

номер ревизии — это слово устанавливается в нуль при создании файла, его содержимое увеличивается на 1 всякий раз, когда файл закрывается после корректировки;

дата ревизии — 7 байт отводятся для хранения даты последней ревизии файла, дата ревизии хранится в символьном коде в формате: день — месяц — год (2 байта — 3 байта—2 байта—соответственно), имеет смысл, только если номер ревизии не равен нулю;

время ревизии — 6 байт хранят время последней ревизии; оно записывается в символьном коде в формате: часы — минуты — секунды (по 2 байта);

дата создания — дата создания файла хранится в семибайтном поле в том же формате, что и дата ревизии;

время создания — время создания файла находится в шестибайтном поле, как и время ревизии;

дата уничтожения — дата, когда истекает срок хранения файла на томе, формат этой даты такой же, как формат даты ревизии.

3.3. ОБЛАСТЬ ПЛАНИРОВАНИЯ

Область планирования содержит информацию, необходимую для преобразования виртуального номера блока в логический номер блока. Это делается с помощью указателей, каждый из которых указывает на непрерывную область. Указатель извлечения файла содержит поле счетчика и поле логического номера блока. Поле счетчика определяет число блоков, содержащихся в непрерывной области, которую описывает данный указатель. Поле логического номера блока указывает логический номер первого блока определяемой области, которую описывает указатель (рис. П.1).

Значение N в поле счетчика означает, что распределено (N+1) блоков, начиная с блока с указанным номером.

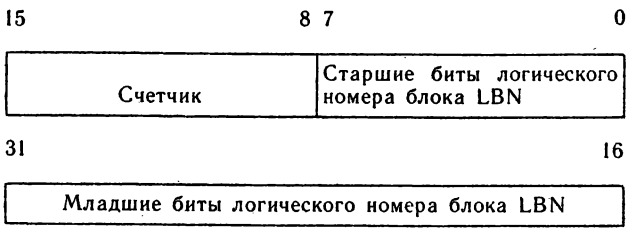


Рис. П.1. Формат указателя извлечения файла

Область планирования обычно имеет пространство для 102 указателей извлечения. Следовательно, можно описать до 102 отдельных сегментов на диске, или до 26112 блоков, если файл является непрерывным. Если требуется больше указателей извлечения, поскольку файл слишком большой или состоит из слишком большого числа отдельных сегментов, то заголовки расширения распределяются, чтобы получить дополнительные указатели извлечения. Заголовки расширения распределяются внутри индексного файла. Они идентифицируются по номеру файла и по последовательному номеру файла, как и другие заголовки файлов; однако заголовки файлов расширения не появляются ни в одном каталоге.

Ненулевое значение в поле номера файла расширения области планирования указывает, что существует заголовок расширения. Заголовок расширения идентифицируется по номеру файла расширения и по последовательному номеру файла расширения. Номер сегмента расширения предназначен для нумерации заголовка файла, для первого сегмента используется номер 0.

Заголовки расширения файла содержат область заголовка и область идентификации, скопированные с первого заголовка, когда он возникает при создании первого расширения. Если первый заголовок файла модифицируется, заголовки расширения не корректируются. Заголовки расширения создаются и управляются процессором управления файлами FCP, когда они необходимы; об их использовании пользователь не знает

ПРИЛОЖЕНИЕ 4

ФАЙЛОВАЯ СТРУКТУРА НА МАГНИТНОЙ ЛЕНТЕ СО СТАНДАРТНЫМИ МЕТКАМИ

Это приложение определяет стандарты для магнитной ленты в системе ОСРВ. В файловой структуре ОСРВ на магнитной ленте возможность работы с переходящими записями не обеспечивается.

4.1. МЕТКИ ТОМА И ФАЙЛА

В табл. П.12, П.13, П.14 и П.15 представлены форматы метки тома и меток заголовка файла.

4.1.1. Формат метки тома

Таблица П.12

Формат метки тома

Позиция символа	Имя поля	Длина поля (в байтах)	Содержимое поля
1—3	Идентификатор метки	3	VOL
4	Номер метки	1	1
5—10	Идентификатор тома	6	Метка тома, любые стандартные символы метки тома Стандартные символы метки тома включают латинские символы от A до Z, цифры от 0 до 9 и специальные символы
11	Доступность	1	Любой стандартный символ метки тома Пробел означает отсутствие ограничений
12—37	Резервируются	26	Пробелы
38—51	Идентификация владельца	14	Содержимое этого поля зависит от системы, под управлением которой том создан, и используется с целью защиты тома
52—79	Резервируются	28	Пробелы
80	Указатель стандартной метки	1	1

Поле идентификации владельца делится на четыре области:
идентификация системы (позиции 38—40);
код защиты тома (позиции 41—44);
код идентификации пользователя (позиции 45—50),
пустой знак из одного пробела (позиция 51)

Поле идентификации системы содержит следующую последовательность символов: D%B. Это означает, что том создан под управлением системы OCPB.

Код защиты тома (позиции 41—44) определяет защиту от доступа на томе для четырех классов пользователей. Каждый класс пользователей имеет привилегии доступа, заданные в одной из четырех позиций следующим образом:

41—системный пользователь;
42—владелец (номер группы и члена должны совпадать);
43—групповой пользователь (номер группы должен совпадать),
44—общий пользователь (любой пользователь, не включенный в три класса, указанные выше)

Один из приведенных ниже кодов доступа может быть задан для каждой позиции символа:

0—нет доступа;
1—только чтение;
2—доступ для расширения (добавления),
3—доступ для чтения/расширения;
4—общий доступ.

Код UIC указан в позициях 45—50. Первые три символа — номер группы (восьмеричный), следующие три символа — номер члена

(восьмеричный), последний символ в поле идентификации владельца— пробел

Пример поля идентификации владельца

D%B1410051102_

где D%B — идентификация системы (файл был создан в ОСРВ),

_ — означает пробел

Системный и групповой пользователи имеют доступ для чтения Владелец имеет общий доступ. Все другие категории доступа отвергаются. UIC—[051 102]

4.1.2. Метки пользователя тома

Метки пользователя (UHL, UTL) тома никогда не пишутся и не сообщаются пользователю. Если метки есть, то они пропускаются.

4.1.3. Метки заголовка файла

Для создания метки заголовка файла необходима следующая информация:

для задания идентификатора файла на магнитной ленте со стандартными метками используются соглашения об именах файлов в файловой структуре ОСРВ (используются символы, допустимые в RADIX50),

символ ( ), употребляемый в идентификаторе файла, имеет фиксированное значение в файловой структуре ОСРВ;

из 17 байт, зарезервированных для идентификатора файла в метке HDR1, в файловой структуре ОСРВ используется максимум 13: 9— собственно для имени, 1—для точки, 3—для типа;

для данных, передаваемых между системой ОСРВ и другими системами, должны соблюдаться соглашения, перечисленные ниже.

Таблица П.13

Метка заголовка файла HDR1

Позиция символа	Имя поля	Длина поля (в байтах)	Содержимое поля
1—3	Идентификатор метки	3	HDR
4	Номер метки	1	1
5—21	Идентификатор файла	17	Любые стандартные символы метки тома (подробно описаны в табл. П.12)
22—27	Идентификатор набора файлов	6	Идентификатор тома (первого тома в наборе томов)
28—31	Номер секции	4	Цифровые символы. Это поле равно 0001 для файла, расположенного на одном томе, и увеличивается на 1 для каждого следующего тома, используемого для этого файла
32—35	Последовательный номер файла	4	Номер файла, указывающий относительное положение файла в наборе файлов на томе Этот номер всегда равен 0001 для единственного файла в наборе файлов

Продолжение

Позиция символа	Имя поля	Длина поля (в байтах)	Содержимое поля
36—39	Номер поколения	4	Цифровые символы
40—41	Номер варианта	2	Цифровые символы
42—47	Дата создания	6	__YYDDD (__означает пробел) или __00000, если даты нет
48—53	Дата уничтожения	6	Формат такой же, как и для даты создания
54	Доступность	1	Пробел
55—60	Счетчик блоков	6	000000
61—73	Код системы	13	__CMFILEOC__ (__означает пробел)
74—80	Резервируется	7	Пробелы

Таблица П.14

Метка заголовка файла HDR2

Позиция символа	Имя поля	Длина поля (в байтах)	Содержимое поля
1—3	Идентификатор метки	3	HDR
4	Номер метки	1	2
5	Формат записи	1	F — фиксированная длина. D — переменная длина. S — сцепленная запись. U — неопределенная длина
6—10	Длина блока	5	Цифровые символы
11—15	Длина записи	5	Цифровые символы
16—50	Системная информация	35	Позиции от 16 до 36— пробелы. Позиция 37 определяет управление кареткой и может иметь одно из следующих значений: A — первый байт записи содержит символ управления кареткой для Фортрана; пробел — коды LF/CR должны быть вставлены между записями; M — запись содержит все виды информации управления
51—52	Смещение в буфере	2	Цифровые символы. На лентах, созданных в ОСРВ, равно 00. Не используется системой ОСРВ при вводе
53—80	Резервируется	28	Пробелы

Таблица П 15

Метка заголовка файла HDR3

Позиция символа	Имя поля	Длина поля (в байтах)	Содержимое поля
1—3	Идентификатор метки	3	HDR
4	Номер метки	1	3
5—68	Системная информация	64	Атрибуты файла, заданные во время его создания. Каждый из 32 байт атрибутов файла пользователя расширяется в два шестнадцатеричных символа
69—80	Резервируется	12	Пробелы

Файловый процессор ОСРВ для магнитной ленты обрабатывает идентификатор файла, как описано ниже. Однако, если имя файла заключено в кавычки, оно обрабатывается, как имя файла для магнитных лент со стандартными метками, все стандартные символы метки тома допустимы, все 17 позиций могут быть использованы, и единственное преобразование, которое имеет место — это преобразование кодов русских символов в латинские и преобразование всех символов, не являющихся стандартными символами метки тома,—в кавычки.

Последовательность шагов, приведенная ниже, описывает обработку идентификатора файла в файловой структуре ОСРВ.

Шаг 1 Из символьного кода в код RAD50 преобразуются только первые десять символов; преобразование продолжается до возникновения одного из следующих условий: ошибка преобразования; преобразовано 9 символов; встретилась точка (.),

Шаг 2 Если встретилась точка, следующие три символа после нее преобразуются и трактуются как тип файла. Если преобразованы все девять символов или встретилась ошибка, то следующий символ проверяется на точку Если этот символ — точка, он пропускается, следующие три символа преобразуются и трактуются как тип файла;

Шаг 3. Номер версии получается из номера поколения и номера варианта поколения по следующей формуле:

(номер поколения — 1) * 100 + номер варианта + 1

Если в результате вычисления получится неправильный номер версии, в качестве номера версии берется 1

Идентификатор файла в выводном файле имеет следующие значения:

имя файла размещается в первых позициях в поле идентификатора файла. Оно может занимать до девяти позиций, далее следует точка;

тип файла (до трех символов) размещается после точки, оставшаяся часть поля заполняется пробелами;

номер версии размещается в полях номера поколения и номера варианта поколения.

4.1.4. Метки конца тома (*EOV*)

Метки конца тома идентичны меткам заголовка файла со следующими исключениями:

позиции символов от 1 до 4 содержат EOV1 вместо HDR1,

поле счетчика блоков содержит число записей в последней секции файла на томе.

4.1.5. Конечные метки

Конечные метки или метки конца файла (EOF) идентичны меткам заголовка со следующими исключениями:

позиции с 1 до 4 содержат EOF1 и EOF2 вместо HDR1 и HDR2;

счетчик блоков содержит количество блоков данных в файле.

4.2. ФАЙЛОВАЯ СТРУКТУРА НА МАГНИТНОЙ ЛЕНТЕ

Ниже рассматривается структура различных комбинаций типов файлов и томов, которая создается файловым процессором на магнитной ленте. Многотомные файлы могут обрабатываться файловым процессором.

Минимальные размеры блока и записи фиксированной длины 18 байт Максимальный размер блока — 8192 байта

Если метка HDR2 отсутствует, файловый процессор за тип данных будет принимать записи фиксированной длины (F), значение длины блока и длины записи берется по умолчанию 512 байт Умолчание можно изменить с помощью программы MAG

При описании файловой структуры на магнитной ленте используются следующие обозначения.

*—маркер ленты EOF,

BOT — начало ленты,

EOT — конец ленты,

, — физический разделитель записей.

Один однотомный файл:

BOT, VOL1, HDR1, HDR2, HDR3* —данные — — *EOF1, EOF2**

Один многотомный файл:

BOT, VOL1, HDR1, HDR2, HDR3* — данные —*EOV1, EOV2**

BOT, VOL1, HDR1, HDR2, HDR3* — данные — —*EOF1, EOF2**

Несколько однотомных файлов:

BOT, VOL1, HDR1, HDR2, HDR3* — —данные — — —*EOF1, EOF2*, HDR1, HDR2, HDR3* — — —данные — — —*EOF1, EOF2**

Несколько многотомных файлов:

BOT, VOL1, HDR1, HDR2, HDR3* — — —данные — — —*EOF1 EOF2*, HDR1, HDR2, HDR3* — — —данные — — —*EOV1, EOV2**

BOT, VOL1, HDR1, HDR2, HDR3* — — —данные — — —*EOF1, OF2*, HDR1, HDR2, HDR3* — — —данные — — —*EOF1, EOF2**

4.3. Обработка конца ленты

Конец ленты обрабатывается автоматически файловым процессором. Файл будет продолжен на следующем томе, смонтированном или монтируемым по запросу Запрос о монтировании следующего тома печатается на устройстве CO: (псевдоустройство вывода на консольный терминал)

4.4. СТАНДАРТНЫЙ БЛОК ЗАГОЛОВКА ФАЙЛА НА МАГНИТНОЙ ЛЕНТЕ (*FCS* — СОВМЕСТИМЫЙ)

На рис. П.2. приведен формат блока заголовка файла, который возвращается по запросу чтения атрибутов заголовка файла на магнитной ленте. Блок заголовка файла магнитной ленты строится из данных полей меток на ленте.

H.MPOF	СМЕЩЕНИЕ 1 СМЕШЕНИЕ 2	H.IDOF
Область заголовка	Последовательный номер файла	H.FNUM
	Номер секции файла	H.FSEQ
	Уровень структуры = 401 (8)	H.FLEV
	UIC (для тома)	H.FOWN = = H.PROG
	Код защиты (для тома)	H.FPRO
	Атрибут записи и код типа записи	H.UFAT
	Размер записи в байтах	
	N слов нулей	
Область идентификации	Имя файла в RADIX-50	X + I.FNAM
	Тип файла в RADIX-50	X + I.FTYP
	Номер версии файла	X + I.FVER
	Нули (дата и время ревизии)	X + I.RVNO
	Дата и время создания (000000)	X + I.CRDT
	Дата уничтожения	X + I.EXDT
	Пустые байты (нули)	X + 47
	Копия метки HDR1	X + 50
	Копия метки HDR2 (если первый байт метки = 0, метка не существует)	X + 130
Область планирования	Пустая область планирования, нули (10 байт)	X + 210

Рис. П.2. Формат блока заголовка файла на магнитной ленте:
смещение 1 означает смещение до области планирования, а смещение 2 — до области идентификации, X = смещению 2×2

ПРИЛОЖЕНИЕ 5

БЛОК ОПИСАНИЯ ФАЙЛА *FDB*

Блок описания файла FDB содержит информацию, которая используется системой управления файлами FCS и процессором управления файлами FCP.

Имена смещений внутри блока описания файла могут быть определены как локальные или глобальные:

FDOF⊙L — как локальные

FDOFF⊙ DEF⊙L как локальные
FDOFF⊙ DEF⊙G как глобальные

Примечание При ссылках на поля блока FDB важно использовать символические имена смещений, а не действительные адреса этих полей Значения смещений полей могут быть различными в разных версиях системы, в то время как имена этих смещений остаются постоянными.

Формат блока описания файла FDB приведен на рис. П.3. Определения полей блока даны в табл. П.16.

0	F RATT и F RTYP	Секция атрибутов файла
2	F RSIZ	
4	F HIBK	
10	F.EFBK	
14	F FFBY	Секция доступа к записи или блоку
16	F RCTL и F RACC	
20	F BKDS или F URBD	
24	F NRBD или F BKST и F.BKDN	
30	F OVBS или F NREC	
32	F EOBB	
34	F RCNM или F CNTG и F STBK	
40	F ALOC	Секция открытия файла
42	F FACC и F LUN	
44	F DSPT	
46	F DFNB	
50	F BKPI и F EFN или F BKEF	Секция блочных буферов
52	F ERR+1 и F ERR	
54	F MBC1 и F MBCT	
56	F BGBC и F MBFG	
60	F VBSZ	
62	F BBFS	
64	F BKVB или F VBN	
70	F BDB	
72	F SPDV	
74	F CHR и F SPUN	
76	F ACTL	
100	F SEQN	
102	F FNB	Секция блока имени файла

Рис. П.3. Формат блока FDB

Определение полей блока *FDB*

Таблица П.16

Поле	Размер (в байтах)	Содержимое
F RTYP	1	Байт типа записи. Этот байт устанавливается для указания типа записи в файле следующим образом: F RTYP=1 указывает фиксированную длину записи (R FIX), F RTYP=2— переменную длину записи (R VAR), F RTYP=3— последовательные записи (R SEQ)

Продолжение

Поле	Размер (в байтах)	Содержимое
F.RATT	1	Байт атрибутов записи. В битах 0—3 устанавливаются атрибуты записи: бит 0=1 указывает, что первый байт записи должен содержать символ управления кареткой (FD.FTN), в противном случае — этот бит равен 0; бит 1=1 указывает для устройств с управляемой кареткой, что символ LF должен быть выведен после вывода строки (FD.CR), в противном случае этот бит равен 0; бит 2 не используется; бит 3=1 указывает, что записи не могут пересекать границу блока (FD.BLK); в противном случае этот бит равен 0
F.RSIZ	2	Слово размера записи. Это поле содержит размер записей фиксированной длины или размер наибольшей записи в файле записей переменной длины
F.HIBK	4	Указывает максимальный виртуальный номер блока, распределенного для файла
F.EFBK	4	Содержит номер последнего блока файла
F.FFBY	2	Указывает первый свободный байт в последнем блоке или максимальный размер блока для МЛ
F RACC	1	Байт доступа к записи. Биты 0—3 этого байта определяют режим доступа к записи: бит 0=1— режим READ⊙/WRITE⊙ (FD.RWM); бит 0=0— режим GET⊙/PUT⊙; бит 1=1— режим прямого доступа (FD.RAN) для ввода-вывода записей GET⊙/PUT⊙; бит 1=0— режим последовательного доступа; бит 2=1— режим указания места (FD PLC) для ввода-вывода записей GET⊙/PUT⊙, бит 2=0— режим пересылки; бит 3=1— операция PUT⊙ в режиме последовательного доступа выполняется без усечения файла (FD.INS); бит 3=0— операция PUT⊙ в режиме последовательного доступа выполняет усечение файла
F RCTL	1	Байт характеристик устройства. Биты 0—5 определяют характеристики устройства, связанного с файлом: бит 0=1 указывает устройство, ориентированное на записи (FD.REC), например, телетайп или печатающее устройство; бит 0=0— устройство, ориентированное на блоки, например, диск; бит 1=1 — устройство с управлением кареткой (FD.CCL), в противном случае 0; бит 2=1 — устройство телепринтер (FD.TTY), в противном случае 0; бит 3=1— устройство с каталогами (FD.DIR), в противном случае 0; бит 4=1— устройство с единственным каталогом (FD.SDI). Используется главный каталог MFD, а пользовательские каталоги UFD не используются; бит 5=1— устройство, ориентированное на блоки с последовательным методом доступа. Устройство, ориентированное на записи, всегда является устройством с последовательным методом доступа, поэтому этот бит для таких устройств не устанавливается

Поле	Размер (в байтах)	Содержимое
F.BKDS или F.URBD	4	Содержит описатель буфера для ввода-вывода блоков Содержит описатель пользовательского буфера записей
F.NRBD или F.BKST	4	Содержит описатель буфера следующей записи Содержит адрес блока состояния ввода-вывода блоков
F.BKDN	2	Содержит адрес обслуживающей программы AST для ввода-вывода блоков
F.OVBS	2	Перекрываемый размер блочного буфера. Это поле имеет смысл только до открытия файла
F.NREC	2	Содержит адрес следующей записи в блоке
F.EOBB	2	Содержит значение, определяющее конец блочного буфера
F.RCNM	4	Содержит номер записи для операций прямого доступа
F.CNTG	2	Содержит число блоков, которые должны быть выделены для файла при его создании. Эта ячейка имеет смысл только до открытия файла. Значение 0 означает, что файл оставляется пустым. Положительное значение означает, что для файла распределяется непрерывная область с указанным числом блоков и файл будет непрерывным. Отрицательное значение означает, что выделяемая область для файла не является непрерывной и файл может быть прерывистым
F.STBK	2	Содержит адрес блока статистики в пользовательской программе
F.ALOC	2	Число блоков, которые должны быть выделены для расширения файла. Это поле имеет смысл только до открытия файла. Положительное значение указывает непрерывное расширение, отрицательное значение — прерывистое
F.LUN	1	Содержит логический номер устройства, связанного с FDB
F.FACC	1	Байт доступа к файлу. Этот байт указывает привилегии доступа к файлу: бит 0=1, если файл доступен только для чтения (FA.RD); бит 1=1, если файл доступен для записи (FA.WRT); бит 2=1, если файл доступен для расширения (FA.EXT); бит 3=1, если создается новый файл (FA.CRE), для существующего файла бит 3=0; бит 4=1, если файл является временным файлом (FA.TMP); бит 5=1, если файл открывается для разделенного доступа (FA.SHR); бит 6=1 при бите 3=0, если к существующему файлу будет добавление (FA.APD), бит 6=1 при бите 3=1, если существующий файл не заменяется на вновь создаваемый файл (FA.NSP)
F.DSPT	2	Содержит указатель блока описания набора данных

Продолжение

Поле	Размер (в байтах)	Содержимое
F.DFNB	2	Содержит указатель блока имени файла по умолчанию
F BKEF или F.EFN	1	Содержит флаг событий ввода-вывода блоков Содержит флаг событий ввода-вывода записей
F BKP1	1	Содержит биты для внутреннего управления
F ERR	1	Байт кода ошибки. Отрицательное значение указывает на наличие ошибки
F ERR+1	1	Используется в комбинации с полем F.ERR; если значение поля F ERR отрицательно, то F ERR+1=0 указывает код ошибки ввода-вывода, а F ERR+1<0— код ошибки в слове состояния директивы
F.MBCT	1	Указывает число буферов для многоуровневой буферизации
F MBC1	1	Указывает число фактически используемых в настоящее время буферов, если употребляется версия системы FCS с многоуровневой буферизацией
F.MBFG	1	Байт флагов многоуровневой буферизации содержит один из следующих флагов: бит 0=1 указывает предварительный ввод (FD.RAH); бит 1=1— довывод (FD WBH)
F BGBC	1	Счетчик больших блочных буферов, если используется версия системы FCS с большими буферами
F VBSZ	2	Размер буфера устройства. Содержит размер виртуального блока (в байтах)
F BBFS	2	Указывает размер блочного буфера
F.BKVB или F VBN	4	Содержит виртуальный номер блока для ввода-вывода блоков Содержит виртуальный номер блока
F BDB	2	Содержит адрес описателя блочного буфера. Это поле всегда содержит ненулевое значение, если файл открыт, и нуль, если файл закрыт
F.SPDV	2	Резервируется для последующих применений
F.SPUN	1	Резервируется для последующих применений
F.CNR	1	Байт характеристик тома: бит 0=1 указывает магнитную ленту со стандартной меткой формата D или F
F.ACTL	2	Младший байт этого слова указывает число указателей извлечения, которые должны быть использованы для файла. Старший байт используется следующим образом: бит 15=1 указывает, что управляющая информация должна быть взята из F.ACTL(FA.ENB), бит 12=0— вызывает позиционирование на конец тома на магнитной ленте при открытии или закрытии; бит 12=1— вызывает позиционирование на магнитной ленте, чтобы пропустить недавно закрытые файлы, если открывается следующий файл (FA.POS); бит 11=1— перемотка магнитной ленты на начало при откры-

Продолжение

Поле	Размер (в байтах)	Содержимое
		тии и закрытии (FA.RWD), бит 9=1 — файл не должен блокироваться, если он неправильно закрыт (FA.DLK)
F.SEQN	2	Содержит последовательный номер для последовательных записей
F.FNB	2	Начальный адрес части блока FDB, называемой блоком имени файла

ПРИЛОЖЕНИЕ 6

БЛОК ИМЕНИ ФАЙЛА

Формат блока имени файла приведен в табл. П.17.

Имена полей в блоке имени файла могут быть определены или локально, или глобально: NBOF⊙L — как локальные; NBOFF⊙DEF⊙L — как локальные, NBOFF⊙ DEF⊙G — как глобальные.

Таблица П.17

Определение полей блока имени файла

Поле	Размер (в байтах)	Содержимое
N.FID	6	Поле идентификационного номера файла
N.FNAM	6	Поле имени файла; указывается как девять знаков в формате RADIX-50
N.FTYP	2	Поле типа файла: указывается как три знака в формате RADIX-50
N.FVER	2	Поле номера версии файла (двоичный)
N.STAT	2	Слово состояния блока имени файла (табл. П.18)
N.NEXT	2	Информация для следующей операции .FIND
N.DID	6	Поле идентификационного номера файла каталога
N.DVNM	2	Поле имени устройства в символьном коде
N.UNIT	2	Поле номера устройства (в двоичном коде)

Примечание При ссылках на поля блока имени файла лучше использовать символические имена смещений, а не действительные адреса этих полей. Адрес информации внутри блока имени файла может подвергаться изменениям при изменении версии системы, в то время как имена этих смещений остаются постоянными.

Определение битов слова состояния блока имени файла N.STAT в блоке FDB и их значения описаны в табл. П.18.

Поля блока имени файла для магнитной ленты со стандартными метками описаны в табл. П.19.

Таблица П.18

Слово состояния блока имени файла *N.STAT*

Символ	Значение (восьмеричное)	Значение
NB.VER*	1	Устанавливается, если явно указан номер версии файла
NB.TYP*	2	Устанавливается, если явно указан тип файла
NB.NAM*	4	Устанавливается, если явно указано имя файла
NB.SVR*	10	Устанавливается, если номер версии указан звездочкой
NB.STP	20	Устанавливается, если тип файла указан звездочкой
NB.SNM	40	Устанавливается, если имя файла указано звездочкой
NB.DIR*	100	Устанавливается, если явно указана строка кода идентификации UIC
NB.DEV*	200	Устанавливается, если явно указана строка имени устройства
NB.SD1	400	Устанавливается, если номер группы в коде идентификации пользователя (UIC) указан звездочкой
NB.SD2	1000	Устанавливается, если номер члена в коде идентификации пользователя (UIC) указан звездочкой
NB.ANS	2000	Устанавливается, если имя файла для магнитной ленты со стандартными метками

Примечание. Символы, отмеченные знаком звездочка, указывают биты, которые устанавливаются, если соответствующая информация берется из блока описания набора данных, представленных в символьном коде.

Таблица П.19

Определение смещений блока имени файла для магнитной ленты со стандартными метками

Поле	Размер (в байтах)	Содержимое
N.FID	2	Поле идентификационного номера файла
N.ANM1	12	Первые 12 байт имени файла
N.FVER	2	Поле номера версии файла (двоичный)
N.STAT	2	Слово состояния блока имени файла (см. табл. П.18)
N.NEXT	2	Информация для следующей операции .FIND
N.ANM2	6	Остальные байты имени файла
N.DVNM	2	Поле имени устройства (символьный код)
N.UNIT	2	Поле номера устройства (двоичный)

ПОДПРОГРАММЫ СИСТЕМНОЙ БИБЛИОТЕКИ

В этом приложении дано краткое описание всех подпрограмм системной библиотеки. Детальное изложение каждой подпрограммы и требования к их использованию приведены в документации по системе ОСРВ.

7.1. ПОДПРОГРАММЫ УПРАВЛЕНИЯ РЕГИСТРАМИ

⊙*SAVAL — подпрограмма сохранения всех регистров.* Ее функции — сохранение/восстановление R0—R5.

Оператор вызова:

JSR PC, ⊙SAVAL

⊙*SAVRG — подпрограмма сохранения регистров 3—5.* Ее функции — сохранение/восстановление R3—R5.

Оператор вызова:

JSR R5, ⊙SAVRG

⊙*SAVVR — подпрограмма сохранения регистров 0—2.* Ее функции—сохранение/восстановление R0—R2.

Оператор вызова:

JSR R2, ⊙SAVVR

⊙*SAVR1 — подпрограмма сохранения регистров 1—5.* Ее функции — сохранение/восстановление R1—R5.

Оператор вызова:

JSR R5, .SAVR1

7.2. АРИФМЕТИЧЕСКИЕ ПОДПРОГРАММЫ

⊙*MUL — подпрограмма умножения целых*

Входные аргументы:

R0 — множитель;

R1 — множимое.

Оператор вызова:

CALL ⊙MUL

Выходные аргументы:

R0 — старшие разряды результата;

R1 — младшие разряды результата;

R2—R5 — сохраняются

⊙*DIV — подпрограмма деления целых*

Входные аргументы:

R0 — делимое;

R1 — делитель.

Оператор вызова:

CALL ⊙DIV

Выходные аргументы:

R0 — результат;

R1 — остаток;

R2—R5 — сохраняются.

⊙*DMUL — подпрограмма умножения с двойной точностью*

Входные аргументы:

R0 — множитель;

R2 — старшие разряды множимого;

R3 — младшие разряды множимого.
Оператор вызова
CALL ⊙DMUL
Выходные аргументы:
R0 — старшие разряды результата;
R1 — младшие разряды результата;
Бит C=0
R4—R5 — сохраняются;
R2—R3 — изменяются.

⊙DDIV — подпрограмма деления с двойной точностью

Входные аргументы:
R0 — делитель без знака;
R1 — старшие разряды делимого;
R2 — младшие разряды делимого.
Оператор вызова
CALL ⊙DDIV
Выходные аргументы
R0 — остаток;
R1 — старшие разряды результата частного;
R2— младшие разряды результата частного;
R3—R5 — регистры сохраняются.

Примечание. Арифметические подпрограммы допускают беззнаковые входные данные, в результате получаются беззнаковые выходные аргументы.

7.3. ПОДПРОГРАММЫ ПРЕОБРАЗОВАНИЯ ДАННЫХ ИЗ СИМВОЛЬНОГО ФОРМАТА ВО ВНУТРЕННИЙ

.DD2CT — подпрограмма преобразования десятичного числа в двухсловный формат.

Входные аргументы:
R3 — адрес двухсловного поля, в котором должно храниться преобразованное число;
R4 — число символов в преобразуемой строке;
R5 — адрес преобразуемой строки символов.
Оператор вызова
CALL .DD2CT
Выходные аргументы:
успешное преобразование:
R3 — адрес двухсловного поля преобразованного числа (1 слово — старшие разряды поля; 2 слово— младшие разряды поля);
бит C=0
неуспешное преобразование:
бит C=1;
все регистры сохраняются.

.OD2CT — подпрограмма преобразования восьмеричного числа в двухсловный двоичный формат

Входные аргументы:
R3 — адрес двухсловного поля, в котором хранится преобразованное число;
R4 — число символов в преобразуемой строке;
R5 — адрес преобразуемой строки символов.
Оператор вызова
CALL .OD2CT

Выходные аргументы:
успешное преобразование:
R3 — адрес двухсловного поля преобразованного числа (1 слово—старшие разряды поля; 2 слово—младшие разряды поля),
бит C=0
неуспешное преобразование:
бит C=1;
все регистры сохраняются.

⊙*CDTB — подпрограмма преобразования десятичных чисел в двоичные*

Входные аргументы:
R0 — адрес первого байта преобразуемой строки символов.
Оператор вызова:
CALL ⊙CDTB
Выходные аргументы:
R0 — адрес байта, следующего за преобразованной строкой символов;
R1 — преобразованное число;
R2 — ограничительный символ;
R3—R5 — регистры сохраняются.

⊙*COTB —подпрограмма преобразования восьмеричных чисел в двоичные*

Входные аргументы:
R0 — адрес первого байта преобразуемой строки символов.
Оператор вызова:
CALL ⊙COTB
Выходные аргументы:
R0 — адрес байта, следующего за преобразованной строкой символов;
R1 — преобразованное число;
R2 — ограничительный символ;
R3—R5 — регистры сохраняются.

⊙*CAT5 — подпрограмма преобразования символов в символьном коде в формат RADIX-50*

Входные аргументы:
R0 — адрес первого символа в символьном коде в преобразуемой строке;
R1=0 — точка является ограничительным символом при преобразовании;
R1=1 — точка должна восприниматься как допустимый символ при преобразовании в формат RADIX-50
Оператор вызова:
CALL ⊙CAT5
Выходные аргументы:
успешное преобразование
R0 — адрес следующего символа в преобразуемой строке;
R1 — значение, преобразованное в формат RADIX-50;
R2 — ограничительный символ;
бит C=0;
неуспешное преобразование:
R2 — недопустимый символ: вызвал прекращение преобразования
бит C=1;
R3—R5 — регистры сохраняются.

⊙*CAT5B — подпрограмма преобразования символов в символьном коде с пробелами в формат RADIX-50*

Входные аргументы:
R1=0 — точка является ограничительным символом при преобразовании;
R1=1 — точка воспринимается как допустимый символ при преобразовании в формат RADIX-50.
Оператор вызова:
CAL ⊙CAT5B
Выходные аргументы:
успешное завершение
R0 — адрес следующего символа в преобразуемой строке;
R1 — значение, преобразованное в формат RADIX-50;
R2 — ограничительный символ;
бит C=0;
неуспешное завершение
бит C=1;
R3—R5 — регистры сохраняются.

7.4. ПОДПРОГРАММЫ ПРЕОБРАЗОВАНИЯ ДАННЫХ ИЗ ВНУТРЕННЕГО ФОРМАТА В СИМВОЛЬНЫЙ

⊙CBDAT—подпрограмма преобразования даты из двоичного представления в символьное
Входные аргументы:
R0 — начальный адрес области для выходных данных;
R1 — преобразуемая двоичная дата;
R2=0 — подавление ведущих нулей в преобразованной дате;
R2≠0—ведущие нули не должны подавляться
Оператор вызова:
CALL ⊙CBDAT
Выходные аргументы:
преобразованная дата содержится по адресу, указанному в регистре R0;
R0 — адрес байта, следующего в области пользователя за последней сохраненной цифрой даты, полученной после преобразования;
R3—R5 — сохраняются;
R1 — R2 — изменяются.
⊙CBDMG — подпрограмма преобразования двоичного числа в десятичное число
Входные аргументы:
R0 — начальный адрес области, в которую должно быть помещено преобразованное число;
R1 — преобразуемое двоичное число без знака;
R2=0 — подавление ведущих нулей в преобразованном числе;
R2≠0 — ведущие нули не должны подавляться.
Оператор вызова:
CALL ⊙CBDMG
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0 — адрес байта, следующего в области пользователя за последней сохраненной цифрой числа, полученного после преобразования;
R3—R5 — сохраняются;
R1—R2—изменяются.
⊙CBDSG — подпрограмма преобразования двоичного числа в десятичное число со знаком

Входные аргументы:
R0—начальный адрес области, в которую должно быть помещено преобразованное число;
R1—преобразуемое двоичное число;
R2=0—подавление нулей в преобразованном числе;
R2≠0—нули не должны подавляться.
Оператор вызова:
CALL ⊙CBDSG
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0—адрес байта, следующего в области пользователя за последней сохраненной цифрой числа, полученного после преобразования;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙CDDMG — подпрограмма преобразования двоичного числа с двойной точностью в десятичное число

Входные аргументы:
R0—начальный адрес области для преобразованного числа;
R1—адрес двухсловной области, содержащей число с двойной точностью;
R2=0—подавление нулей;
R2≠0—нули не подавляются.
Оператор вызова:
CALL ⊙CDDMG
Выходные аргументы:
успешное преобразование
преобразованное число находится в указанной пользователем области;
неуспешное преобразование
по адресу, указанному пользователем, находится строка, состоящая из «звездочек»;
R0—адрес байта, следующего в области пользователя за последней сохраненной цифрой числа, полученного после преобразования;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙CBOMG—подпрограмма преобразования двоичного числа в восьмеричное без знака

Входные аргументы:
R0—начальный адрес области, в которой должно быть размещено преобразованное число;
R1—преобразуемое число;
R2=0—подавление нулей;
R2≠0—нули не подавляются.
Оператор вызова:
CALL ⊙CBOMG
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0—адрес байта, следующего в области пользователя за преобразованным числом;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙CBOSG — подпрограмма преобразования двоичного числа в восьмеричное число

Входные аргументы:
R0—начальный адрес области, в которой размещается число;

R1 — преобразуемое двоичное число;
R2=0—подавление ведущих нулей;
R2≠0—ведущие нули не подавляются.
Оператор вызова:
CALL ⊙CBOSG
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0—адрес байта, следующего в области пользователя за преобразованным числом;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙CBTMG—подпрограмма преобразования двоичного содержимого байта в восьмеричное число

Входные аргументы:
R0—начальный адрес области, в которую должно быть помещено преобразованное число;
R1—преобразуемый двоичный байт;
R2=0—подавление ведущих нулей;
R2≠0—нули не подавляются.
Оператор вызова:
CALL ⊙CBTMG
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0—адрес байта, следующего в области пользователя за преобразованным байтом;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙CBTA—основная подпрограмма преобразования двоичных чисел в символьный код

Входные аргументы:
R0—начальный адрес области, в которой должно быть размещено преобразованное число в символьном коде;
R1 — преобразуемое двоичное число;
R2 — параметры преобразования;
биты 0—7—содержат основание преобразования в коде RADIX-50;
бит 8: =0—число без знака;
=1—преобразуемое значение со знаком;
бит 9: =0 — ведущие нули подавляются;
=1—ведущие нули не подавляются;
бит 10: =1—заполнение ведущих нулей пробелами;
=0—игнорируется заполнение ведущих нулей пробелами;
биты 11—15: ширина поля—(числовое значение от 1 до 32)
Оператор вызова:
CALL ⊙CBTA
Выходные аргументы:
преобразованное число находится в указанной пользователем области;
R0—адрес байта, следующего в области пользователя за последней размещенной цифрой числа, полученного после преобразования;
R3—R5—сохраняются;
R1—R2—изменяются.

⊙C5TA—подпрограмма преобразования данных из кода RADIX-50 в символьный код

Входные аргументы:
R0—адрес, по которому должен быть размещен выходной аргумент;

R1—преобразуемое слово в формате RADIX-50.

Оператор вызова:

CALL ⊙C5TA

Выходные аргументы:

преобразованное число находится в указанной пользователем области;

R0—адрес байта, следующего в области пользователя за последним размещенным символом;

R3—R5—сохраняются;

R1—R2—изменяются.

7.5. ПОДПРОГРАММЫ ПРЕОБРАЗОВАНИЯ И ФОРМАТИРОВАНИЯ ВЫВОДИМЫХ ДАННЫХ

⊙CVTUC — подпрограмма преобразования строки букв русского алфавита

Входные аргументы:

R0—адрес преобразуемой строки текста;

R1—адрес выходной области для размещения преобразованной строки;

R2—число преобразуемых байтов строки (не должно быть равным нулю).

Оператор вызова:

CALL ⊙CVTUC

Выходные аргументы:

в выходной области — преобразованный текст из символьного кода в буквы латинского алфавита;

R0—R1—указывает на символы, следующие за преобразованной строкой;

R2—изменяется;

R3—R5—не используется.

⊙DAT—подпрограмма преобразования строки даты

Входные аргументы:

R0—адрес выходной области;

R1—адрес входной области.

Оператор вызова:

CALL ⊙DAT

Выходные аргументы:

в выходной области — преобразованная строка даты;

R0—адрес байта, следующего в области пользователя за последним размещенным символом преобразованной даты;

R1—адрес следующего слова по входной области;

R3—R5—сохраняются;

R2—изменяется.

⊙TIM — подпрограмма преобразования времени

Входные аргументы:

R0—адрес выходной области;

R1—адрес входной области;

R2—формат преобразования выходных данных;

=0 или 1—час (чч);

=2—часы: минуты (чч:мм);

=3—часы: минуты: секунды (чч:мм:сс);

=4 или 5—часы: минуты: секунды. тики (чч:мм:сс.т)

Оператор вызова:

CALL ⊙TIM

Выходные аргументы:

преобразованная строка находится в указанной пользователем выходной области;

R0—адрес байта, следующего в выходной области за последним размещенным символом результата преобразования;

R1—адрес следующего слова во входной области;

R2—изменяется;

R3—R5—сохраняются;

R0—R1—принимаются новые значения.

⊙ *EDMSG — подпрограмма редактирования сообщений*

Входные аргументы:

R0—начальный адрес выходной области;

R1—адрес входной строки;

R2—начальный адрес блока аргументов, содержащего преобразуемые двоичные данные адреса строк в символьном коде и адреса ячеек с числами с двойной точностью.

Входные строки задаются с помощью директивы макроассемблера .ASCIZ. Эти строки содержат текст в символьном коде и директивы редактирования в одном из следующих форматов:

%L
%NL
%VL

где N—счетчик повторения (десятичное число), определяющий число повторений операции редактирования;

V—указывает, что счетчик повторения является значением из следующего слова в блоке аргументов задачи;

L—буква, определяющая одну из следующих операций редактирования:

A—пересылка символьной строки;

B—преобразование двоичного байта в восьмеричное число;

D—преобразование двоичного числа в десятичное со знаком;

E — пересылка строки с заменой непечатаемых символов на пробелы;

F — включение перевода формата;

I — включение строки ASCIZ;

M — преобразование двоичного числа в десятичное с подавлением нулей;

N — включение знаков возврата каретки и перевода строки;

O — преобразование двоичного числа в восьмеричное со знаком;

P — преобразование двоичного числа в восьмеричное без подавления нулей;

Q — преобразование двоичного числа в восьмеричное с подавлением нулей;

R — преобразование кода RADIX-50 в символьный код;

S — включение пробелов;

T — преобразование двоичных чисел с двойной точностью в десятичное число;

U — преобразование двоичного числа в десятичное число с двойной точностью без подавления нулей;

X — преобразование имени файла;

Y — преобразование даты;

Z — преобразование времени из двоичного вида в символьный;

< — определение заданной длины;

> — нахождение признака конца поля.

Оператор вызова
CALL ⊙EDMSG
Выходные аргументы
преобразованные и/или сформатированные данные, размещенные в выходной области;
R0—адрес последнего байта в выходной области (этот байт равен нулю),
R1—длина выходной области (число байтов, переданных в выходную область),
R2—адрес следующего аргумента в блоке аргументов;
R3—R5—сохраняются.

7.6. ПОДПРОГРАММЫ УПРАВЛЕНИЯ ДИНАМИЧЕСКОЙ ПАМЯТЬЮ

⊙INIDM—подпрограмма установки динамической памяти в начальное состояние
Входные данные
R0—адрес заголовка списка свободных блоков памяти;
FREEHD: BLKW2—включить в секцию данных.
Оператор вызова
CALL ⊙INIDM
Выходные аргументы
R0—первый адрес задачи;
R1—первый адрес свободной динамической памяти;
R2—размер свободной динамической памяти;
R3—R5—не используются.
⊙RQCB—подпрограмма запроса блока памяти
Входные аргументы
R0—адрес заголовка списка свободных блоков памяти;
R1—размер в байтах запрашиваемого блока памяти.
Оператор вызова
CALL ⊙RQCB
Выходные аргументы
успешное завершение
R0—адрес выделенного блока памяти;
R1—действительный размер блока;
бит C=0;
неуспешное завершение:
бит C=1,
R3—R5—сохраняются;
R2—изменяется.
⊙RLCB — подпрограмма освобождения блоков памяти
Входные аргументы:
R0—адрес заголовка списка свободных областей памяти;
R1—размер в байтах освобождаемого блока памяти;
R2—адрес освобождаемого блока памяти.
Оператор вызова:
CALL ⊙RLCB
Выходные аргументы:
освобожденный блок памяти;
R0—не изменяется;
R1—R2—изменяются;
R3- R5—сохраняются.

7.7. ПОДПРОГРАММЫ УПРАВЛЕНИЯ ВИРТУАЛЬНОЙ ПАМЯТЬЮ

⊙INIVM—инициализации виртуальной памяти.

Входные данные:

определить блок ⊙FRHD и занести в него начальный адрес свободной динамической памяти;

определить 3 глобальных символа;

W⊙KLUN—номер LUN рабочего файла;

W⊙KEXT—размер расширения рабочего файла;

N⊙MPAG—счетчик страниц для ускоренного поиска страниц;

R0—старший адрес свободной динамической памяти.

Оператор вызова:

CALL ⊙INIVM

Выходные аргументы:

успешное завершение

бит C=0 и R0=0;

неуспешное завершение:

бит C=1

R0=—2—рабочий файл не открыт;

R0=—1—файл не отмечен на удаление;

R3—R5—сохраняются;

R0—R2—изменяются.

⊙ALBLK—подпрограмма выделения блока

Входные аргументы:

R1—размер запрашиваемого блока в байтах.

Оператор вызова:

CALL ⊙ALBLK

Выходные аргументы:

успешное завершение:

R0—адрес выделенного блока памяти;

неуспешное завершение

вызывается подпрограмма пользователя ⊙ERMSG;

R3—R5—сохраняются;

R0—R2—изменяются.

⊙GTCOR—подпрограмма выделения блока

Входные аргументы:

R1—размер запрашиваемого блока в байтах.

Оператор вызова:

CALL ⊙GTCOR

Выходные аргументы:

успешное завершение

R0—адрес блока динамической памяти;

бит C=0;

неуспешное завершение:

бит C=1;

R3—R5—сохраняются.

⊙EXTSK—подпрограмма расширения задачи

Входные аргументы:

R1—размер запрашиваемого блока (в байтах)

Оператор вызова:

CALL ⊙EXTSK

Выходные аргументы:

успешное завершение

R1—действительный размер расширения;

бит C=0;

неуспешное завершение

бит C=1;
R2—R5—сохраняются.

⊙ *WRPAG—подпрограмма выгрузки страниц*

Входные аргументы:
R2—адрес страницы в памяти.
Оператор вызова:
CALL ⊙WRPAG
Выходные аргументы:
успешное завершение
бит C=0;
неуспешное завершение
бит C=1;
вызывается подпрограмма пользователя ⊙ERMSG;
R0—R2 — сохраняется.

⊙ALVRT — *подпрограмма выделения виртуальной памяти*

Входные аргументы:
R1—размер запрашиваемого блока (в байтах).
Оператор вызова:
CALL ⊙ALVRT
Выходные аргументы:
успешное завершение
R0—адрес выделенного блока памяти;
R1—адрес выделенного блока на диске;
неуспешное завершение:
вызывается подпрограмма пользователя ⊙ERMSG;
R3—R5—сохраняются;
R2—изменяется.

⊙ *ALSVB—подпрограмма выделения малых виртуальных блоков*

Входные аргументы:
N⊙DLGH=512—в вызывающей программе определить глобальную константу
R1—размер запрашиваемого блока страницы;
R1=0—для выделения большого блока при первом обращении к подпрограмме ⊙ALSVB;
R1<512(10)—для выделения малой страницы.
Оператор вызова:
GALL ⊙ALSVB
Выходные аргументы:
R0—адрес блока динамической памяти;
R1—виртуальный адрес блока;
R3—R5—сохраняются;
R2—изменяются.

⊙ *CVLOK—подпрограмма преобразования адреса и блокировки страниц*

Входные аргументы:
R1—виртуальный адрес.
Оператор вызова:
CALL ⊙CVLOK
Выходные аргументы:
успешное завершение
R0—адрес памяти;
R1—виртуальный адрес;
бит C=0;
неуспешное завершение
бит C=1,
R2—R5—сохраняются.

⊙ *CVRL—подпрограмма преобразования виртуального адреса в действительный*

Входные аргументы:
R1—виртуальный адрес.
Оператор вызова:
CALL ⊙CVRL
Выходные аргументы
R0—адрес памяти;
R3—R5—сохраняются;
R1—не изменяется;
R2—изменяется.

⊙*RDPAG—подпрограмма чтения страницы*

Входные аргументы:
R0—адрес страницы на диске.
Оператор вызова:
CALL ⊙RDPAG
Выходные аргументы:
успешное завершение
бит C=0;
неуспешное завершение:
вызывается подпрограмма пользователя ⊙ERMSG;
R0—R2—сохраняются.

⊙*FNDPG—подпрограмма нахождения страницы*

Входные аргументы:
R1—виртуальный адрес страницы.
Оператор вызова:
CALL ⊙FNDPG
Выходные аргументы.
страница найдена
R0—адрес блока памяти:
бит C=0;
страница не найдена:
бит C=1.

⊙*WRMPG—подпрограмма отметки записей в страницу*

Входные аргументы:
R1—виртуальный адрес в странице.
Оператор вызова:
CALL ⊙WRMPG
Выходные аргументы:
бит C=0, страница отмечена записью;
бит C=1, страница не найдена;
R0—R2—сохраняются.

⊙*LCKPG—подпрограмма блокировки страницы*

Входные аргументы.
R1—виртуальный адрес страницы.
Оператор вызова:
CALL ⊙LCKRG
Выходные аргументы
бит C=0, страница блокирована;
бит C=1, страница не найдена;
R0—R2—регистры сохраняются.

⊙*UNLPG—подпрограмма снятия блокировки страницы*

Входные аргументы
R1—виртуальный адрес страницы.
Оператор вызова
CALL ⊙UNLPG
Выходные аргументы
бит C=0, страница разблокирована;
бит C=1, страница не найдена;
R0—R2—сохраняются.

ПРИЛОЖЕНИЕ 8

ТИПЫ ФАЙЛОВ, ПРИНЯТЫЕ В ОСРВ

BAS — исходная программа на языке Бейсик
CBL — исходная программа на языке Кобол
CLB — библиотека косвенных командных файлов
CMD — косвенный командный файл
COR — файл корректировки SLP
DAT — файл данных
DIR — файл каталога
FTN — исходная программа на языке Фортран
LST — файл листинга
MAC — исходная программа на языке макроассемблера
MAP — файл распределения памяти построителя задач
MLB — макробиблиотека пользователя
OBJ — программа в объектных кодах
ODL — файл описания перекрытий
OLB — библиотека объектных модулей
PAT — файл корректировки, используемый для создания корректируемого объектного модуля
POB — вводной файл корректируемого объектного модуля для программы
SML — системная макробиблиотека
STB — файл таблицы определения символов
SYS — загружаемый образ системы
TMP — временный файл
TSK — файл образа задачи
TXT — текстовый файл
ULB — универсальный библиотечный файл

ПРИЛОЖЕНИЕ 9

УПРАВЛЯЮЩИЕ СИМВОЛЫ ТЕРМИНАЛА

Международное обозначение	Русское обозначение	Название	Кодировка на клавиатуре	Объяснение
<ETX>	КТ	Конец текста	CTRL/C	Вызывает передачу управления программ связи с оператором или CLI, написанному пользователем. Ввод ETX с подчиненного терминала не вызывает реакцию системы
<HT>	ГТ	Горизонтальная табуляция	CTRL/I	Вызывает перемещение маркера к следующей позиции табуляции
<VT>	ВТ	Вертикальная табуляция	CTRL/K	Вызывает вывод на терминал четырех символов LF
<FF>	ПФ	Перевод формата	CTRL/L	Вызывает вывод восьми символов
<DLE>	АР1	Авторегистр 1	CTRL/O	Используется для подавления вывода на терминал

Продолжение

Международное обозначение	Русское обозначение	Название	Кодировка на клавиатуре	Объяснение
<CR>	ВК	Возврат каретки	CR	Вызывает окончание ввода и возврат каретки к началу строки
<LF>	ПС	Перевод строки	LF	Вызывает перемещение маркера на следующую строку в ту же позицию
<DEL>	ЗБ	Забой	DEL RUBOUT	Удаляет последний введенный символ
<DC1>	СУ1	Символ устройства 1	CTRL/Q	После CTRL/S предполагает возобновление вывода, приостановленного по CTRL/S
<DC2>	СУ2	Символ устройства 2	CTRL/R	Ввод CTRL/R перед ограничителем строки вызывает печать этой строки без удаленных символов в новой строке
<DC3>	СУ3	Символ устройства 3	CTRL/S	Вызывает останов вывода на терминал
<NAK>	НЕТ	Отрицание	CTRL/U	Вызывает удаление вводимой строки
<SUB>	ЗМ	Замена	CTRL/Z	Используется для завершения задачи
<ESC>	АР2	Авторегистр 2	CTRL/[	Может использоваться как ограничитель строки. При этом возврат каретки и перевод строки не выполняются

Автозагрузка перекрытий 146
Адрес:
виртуальный 16
физический 16

Библиотека
— резидентная 163
— групповая 152
— объектных модулей 11, 157, 196
Блок:
имени файла по умолчанию 63, 71
описания групповых глобальных флагов (GFB) 36, 37
описания набора данных 63, 66
описания окна (WDB) 29
описания района 50
описания файла (FDB) 63, 68, 247
параметров директивы (DPB) 26
состояния ввода-вывода (IOSB) 230
управления задачей (TCB) 14, 19
управления порождением (OCB) 55
Буферизация 64

Важное событие 22, 41
Вектор прерывания 46
Виртуальное адресное окно 17
Виртуальное адресное пространство 16, 139
Выгружаемость задачи 21, 156, 157
— —; запретить 33
— — разрешить 33
Время диспетчеризации задачи 30, 32

Глобальный символ 161
Дерево перекрытий 142, 143
Дефектный блок 180
Динамическая память 228
Динамическое распределение памяти 228

Директива
— ввода-вывода 65
— управляющей программы 25
Диспетчер памяти 15
Драйвер 12
— загружаемый 208
— резидентный 12

Задача
— вывода системных сообщений (TKTN) 206
— потомок 54
— привилегированная 14
— регистрации ошибок оборудования (ERRLOG) 12
— родитель 54
— установленная 19
Заголовок задачи 135, 157
Заголовок файла 237
Закрытие файла 73

Идентификатор виртуального адресного окна 50
Индексный файл 235
Индикатор автозагрузки 146
Интерпретатор командной строки 60
Код идентификации пользователя (UIC) 94, 166
Командная строка MCR 96
— — обслуживающей программы 166
Косвенный командный файл 121
Ключ
— обслуживающей программы 166
— типа передач 174
— типа формата 174
Круговая диспетчеризация 21

Логический номер устройства (LUN) 92, 47
Логический уровень ввода-вывода 8
Метод доступа
— — последовательный 63
— — прямой 63

Метка конца файла (EOF) 246
— тома 242
Назначение устройств 92
Номер блока
— — виртуальный 64
— — логический 64

Обслуживающая программа 9, 165
Открытие ячейки 189
— файла 71

Перекрытие 138
Подсостояние задачи:
готовности к выполнению 20
блокированное 20
останова 20
Построитель задач (ТКВ) 134
Привилегия доступа к файлу 95, 171
Признак удаления файла 72, 76, 82
Приоритет 20, 162
Программная секция 135
Процессор управления файлами (FCP) 209
Псевдоустройства 91

Раздел 15, 162, 206
— системно-управляемый 15
— управляемый пользователем 15
Размер раздела 15
Разделяемая область 148
— — абсолютная 148
— — позиционно-независимая 149
— — системная 160
Разделяемый доступ к файлу 16, 164
Район: 16, 164
базовый адрес 50
идентификатор 50
Регистр переключателей процессора 34
— верхнего предела памяти 191, 193
— нижнего предела памяти 191, 193
Регистрация ошибок оборудования 12
Регистр смещения 189
Режим передачи данных:
пересылки 64
указание места 64
Резервирование дисков 225

Ручная загрузка 21

Сегмент 137
Секция:
атрибутов файла 68
блочных буферов 69
доступа к блокам 68
открытия файла 69
Система виртуального доступа 6
— управления данными (СУД) 8
— управления файлами (FCS 63
Системный каталог задач (STD) 19, 21
Системные прерывания:
асинхронные 24, 42
синхронные 24, 45
Слово состояния директивы (DSW) 25
Слово состояния (CLI) 60—62
Собственный блок тома 236
Состояние задачи:
бездействия 19
активное 19
Спецификация файла 86, 96
Структура данных 63
Счетчик доступа к флагам 37

Терминалы:
непривилегированные 93
привилегированные 94
присоединенные 94
Точка останова 189

Удаление файла 76, 82
Указатель извлечения 19
Управляющая программа 25
Управление файлами 17
Управление памятью 201

Фаза генерации 209, 211
Файл:
выгрузки 21
дефектных блоков 180
образа задачи 134
определения символов 154
распределения памяти 154
Файловая структура 65, 235
Файловая область памяти (FSR) 63
— начальная установка 70
Флаги событий 23
— групповые глобальные 23
— локальные 23
— общие 23

СОДЕРЖАНИЕ

Введение 8

1

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ОСРВ

1.1. Область применения ОСРВ 5
1.2. Структура, состав и функции ОСРВ 7
1.3. Возможности управляющей программы 14

2

ДИРЕКТИВЫ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

2.1. Введение в управляющую программу 25
2.2. Общие соглашения по описанию форматов макровызовов и вызовов подпрограмм 26
2.3. Группы директив 28

3

СИСТЕМА УПРАВЛЕНИЯ ФАЙЛАМИ ОСРВ

3.1. Директивы управления вводом-выводом 65
3.2. Подпрограммы управления файлами 76
3.3. Обработка командных строк 83

4

КОМАНДЫ ОПЕРАТОРА

4.1. Основные понятия устройства 91
4.2. Программа связи с оператором 96
4.3. Интерпретатор командных строк 118
4.4. Диспетчер неустановленных задач TDX 120
4.5. Косвенные командные файлы 121

5

СОЗДАНИЕ ЗАДАЧ

5.1. Функции построителя задач 134
5.2. Структура задачи 134
5.3. Построение задач с перекрытиями 137
5.4. Разделяемые области 148
5.5. Командная строка ТКВ 154
5.6. Ключи построителя задач 156
5.7. Необязательные параметры 159

6

СИСТЕМНЫЕ ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ

6.1 Общие сведения об обслуживающих программах ОСРВ 165
6.2. Программы обработки файлов 167
6.3. Программы проверки и подготовки носителей 179
6.4. Программа сохранения и восстановления томов 183
6.5. Отладочные программы . 188
6.6. Программа-библиотекарь (LBR) 196

7

ГЕНЕРАЦИЯ ОСРВ

7 1 Параметры генерации системы (фаза 1) 202
7.2. Действия в фазе 2 генерации ОСРВ 209
7.3. Действия в фазе 3 генерации ОСРВ 211
7.4. Система виртуального доступа (СВД) 212

8

ПРОГРАММНЫЕ СРЕДСТВА ПОВЫШЕНИЯ НАДЕЖНОСТИ ФУНКЦИОНИРОВАНИЯ СМ ЭВМ

8.1 Подсистема регистрации ошибок оборудования 216
8.2. Программа-анализатор аварийного состояния системы (CDA) 222
8.3. Подсистема резервирования дисков . . 225
8.4. Средства управления динамической памятью системы 228
Литература 229

ПРИЛОЖЕНИЯ

Приложение 1. Коды ошибок 230
Приложение 2. Формат индексного файла 235
Приложение 3. Формат блока заголовка файла 238
Приложение 4. Файловая структура на магнитной ленте со стандартными метками 241
Приложение 5. Блок описания файла FDB 247
Приложение 6. Блок имени файла 252
Приложение 7 Подпрограммы системной библиотеки 254
Приложение 8. Типы файлов, принятые в ОСРВ 266
Приложение 9. Управляющие символы терминала 266
Предметный указатель 268

Справочное издание

Геннадий Алексеевич Егоров,
Валерий Львович Кароль,
Илья Семенович Мостов и др.

ОПЕРАЦИОННАЯ СИСТЕМА ОСРВ СМ ЭВМ

Зав. редакцией И. Г. Дмитриева
Редакторы Л. Д. Григорьева, Л. А. Табакова
Мл. редактор А. А. Орлова
Худож. редактор С. Л. Витте
Техн. редакторы Г. А. Полякова, И. В. Завгородняя
Корректоры И. П. Елкина и Т. М. Иванова
Переплет художника Б. С. Вехтера

ИБ № 2012

Сдано в набор 8.10.87. Подписано в печать 24.03.87. А04375.
Формат 84×108$^{1}/_{32}$. Бум. типографская № 2. Гарнитура «Литературная». Печать высокая. Усл. печ. л. 14,28. Усл. кр.-отт 14,28. Уч.-изд. л. 16,23. Тираж 26.000 экз. Заказ 2124.
Цена 1 р. 10 к.
Издательство «Финансы и статистика», 101000, Москва, ул. Чернышевского, 7.
Московская типография № 4 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли,
129041, Москва, Б. Переяславская ул., д. 46.